

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

**ЭЛЕКТРОННЫЙ СБОРНИК
ТРУДОВ МОЛОДЫХ СПЕЦИАЛИСТОВ
ПОЛОЦКОГО ГОСУДАРСТВЕННОГО
УНИВЕРСИТЕТА**

Выпуск 20 (90)

ПРОМЫШЛЕННОСТЬ

Информационные технологии

Машиностроение

Химические технологии

Обновляется 1 раз в год

Новополоцк

ПГУ

2017

РЕДАКЦИОННАЯ КОЛЛЕГИЯ

Главный редактор:
проф., д.т.н. Д.Н. Лазовский

Заместитель главного редактора:
доц., к.т.н. Д.О. Глухов

Председатель ред. совета проф., д.т.н. В.П. Иванов
Зам. председателя ред. совета проф., д.т.н. Н.Н. Попок

Информационные технологии:

доц., к.т.н. В.Ф. Янушкевич
доц., к.т.н. Р.П. Богуш
доц., к.ф.-м.н. С.А. Вабищевич
доц., к.ф.-м.н. А.А. Козлов

Машиностроение:

доц., к.т.н. А.В. Дудан
доц., к.т.н. Т.В. Вигерина
доц., к.т.н. А.М. Долгих
доц., к.т.н. В.Э. Завистовский

Химическая техника и технологии:

доц., к.т.н. А.В. Митинов
доц., х.т.н. С.Ф. Якубовский
доц., к.пед.н. И.В. Буряя
доц., к.т.н. А.А. Ермак

ЭЛЕКТРОННЫЙ СБОРНИК ТРУДОВ МОЛОДЫХ СПЕЦИАЛИСТОВ ПОЛОЦКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА [Электронный ресурс]. – Новополоцк : Полоцкий государственный университет, 2017. Вып. 20 (90). Промышленность. – 1 электрон. опт. диск.

Издается с 2014 года (в печатном виде – с 2004 г.)

Сборник включен в Государственный регистр информационного ресурса. Регистрационное свидетельство № 3201404563 от 30.11.2014 г.

В сборнике представлены труды молодых специалистов Полоцкого государственного университета по направлениям прикладных исследований, проводимых в университете: информационные технологии, машиностроение, химическая техника и технологии.

Предназначен для работников науки, образования и производства, будет полезен студентам, магистрантам и аспирантам университетов.

Учредитель: учреждение образования «Полоцкий государственный университет»
211440, ул. Блохина, 29, г. Новополоцк, Беларусь
тел. 8 (0214) 39 40 46, e-mail: vestnik@psu.by

Компьютерный дизайн *М. С. Мухоморовой*

Техническое редактирование и компьютерная верстка *Р. Н. Авласенок*

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.[822+823]

СЕМАНТИЧЕСКИЕ ФРЕЙМЫ: КЛАССИФИКАТОРЫ И КВАЛИФИКАТОРЫ

А.И. ДУНЧЕНКО

(Представлено: Д.В. ПЯТКИН)

Рассматривается концепция представления знаний в интеллектуальных компьютерных системах. Раскрывается сущность новой модели, основанной на семантических фреймах и ее связь с уже существующими решениями. Описываются отличительные особенности – классификаторы и квалификаторы, которые являются дополнениями к базовой фреймовой модели.

Введение. При создании систем искусственного интеллекта ключевым вопросом является выбор способа представления и обработки информации об объектах и явлениях реального мира применительно к возможностям современных ЭВМ. В зависимости от типа решаемой задачи, существует два подхода к этой проблеме. *Первый* – это использование искусственных нейронных сетей (ИНС) [1]. ИНС – это аппаратно или программно реализованная сеть, работа которой основана на тех же принципах, что и мозг высших представителей фауны. Каждая такая ИНС получается путем комбинирования множества различных нейронов, реализующих ту или иную математическую функцию. Недостатком нейронных сетей является то, что они требуют длительного обучения и являются ресурсоемким решением. Второй подход заключается в *применении специфической нотации*. Традиционными представителями данной группы являются: *правила* (выражения IF-THEN-ELSE), *предикаты* [2], *семантические сети* [3], *фреймы* [4].

Ни одно из перечисленных решений не является универсальным: все они обладают как определенными преимуществами друг перед другом, так и некоторыми недостатками. Поэтому выбор той или иной нотации сильно зависит от конкретно поставленной перед разработчиками задачи.

Семантические фреймы (СФ) – это попытка создания новой модели представления знаний, в основе которой лежит все та же концепция фреймов, только дополненная элементами из семантических сетей, предикатов и нечеткой логики.

В модели СФ отдельно взятый слот или даже целый фрейм может выражать *семантическое отношение* между связанными элементами. Для этой цели служат *классификаторы*.

Классификаторы

Классификатор – это символическая константа (в текущей спецификации это так; в будущих версиях вполне возможно, что константы будут заменены на классы, методы которых будут определять правила управления отношениями), выражающее характеристику своего операнда в наиболее обобщенной форме. То есть при определении перечня стандартных классификаторов делался упор на минимализм: из ряда близких по смыслу терминов выбирался тот, который имеет более абстрактное значение, включающее в себя множество смежных понятий. Каждый слот или фрейм может иметь несколько не противоречащих друг другу классификаторов, правила, назначения которых определяются спецификацией и приведены в таблицах 1–7.

Таблица 1. – Классификаторы фреймов

Классификатор	Описание
<i>set</i>	Коллекция связанных сущностей: сфера, группа, пространство имен, пакет и т.п.
<i>class</i>	То же, что и в ООП.
<i>prototype</i>	То же, что и в ООП.
<i>enumeration</i>	Перечисление символических констант (аналогично <i>enum</i> в С). Например – <i>сезон: зима, весна, лето, осень</i> .
<i>relation</i>	Множество тесно связанных сущностей. Слоты содержат объекты, связанные данным отношением.
<i>activity</i>	Используется для описания процедурных знаний. Слоты содержат участников (затрагиваемые объекты) и характеристики деятельности (события, явления и т.п.).
<i>object</i>	Экземпляр некоторого класса/прототипа или анонимный объект без определенного типа.

Таблица 2. – Модификаторы

Классификатор	Описание
<i>fixed</i>	Не позволяет добавлять/удалять слот из фрейма или изменять значение слота.
<i>not-inherited</i>	Фреймы, помеченные данным классификатором, не могут наследоваться (<i>class, enumeration</i>). Слоты с таким классификатором не наследуются.
<i>shared</i>	Экземпляры класса вместо копии получают ссылку на слот.
<i>attribute</i>	Анонимный слот. Значение слота содержит некоторую обобщенную характеристику (прилагательное, наречие, числительное).

Таблица 3. – Классификаторы слотов *set*-фреймов

Классификатор	Описание
<i>includes</i>	Безымянный слот, содержащий ссылку на вложенное множество.
	Слоты без перечисленных выше классификаторов содержат обычные элементы множества.

Таблица 4. – Классификаторы слотов *class*-фреймов

Классификатор	Описание
<i>extends</i>	Безымянная ссылка на базовый (расширяемый) класс.
<i>includes</i>	Безымянная ссылка на вложенный класс.
<i>activity</i>	Слот описывает деятельность, производимую экземпляром класса, или событие, на которое он может реагировать.
	Слоты без перечисленных выше классификаторов содержат поля, копии которых будут назначены всем экземплярам класса.

Таблица 5 – Классификаторы слотов *enumeration*-фреймов

Классификатор	Описание
<i>extends</i>	Безымянная ссылка на базовое (расширяемое) перечисление.
<i>activity</i>	Слот описывает деятельность, производимую элементом перечисления, или событие, на которое он может реагировать.
	Слоты без перечисленных выше классификаторов содержат элементы перечисления, а их имена являются идентификаторами этих значений.

Таблица 6. – Классификаторы слотов *relation*-фреймов

Классификатор	Описание
<i>extends</i>	Безымянная ссылка на базовое (расширяемое) отношение.
<i>activity</i>	Слот описывает деятельность, подразумеваемую данным отношением, или событие, которое может наступить в определенный момент.
	Слоты без перечисленных выше классификаторов содержат элементы, связанные отношением (являющиеся активными или пассивными участниками).

Таблица 7. – Классификаторы слотов *activity*-фреймов

Классификатор	Описание
<i>in</i>	Входной параметр (аргумент).
<i>out</i>	Выходной параметр (результат).
<i>in activity</i>	Деятельность, которая может инициировать данную активность (причина).
<i>out activity</i>	Деятельность, которая может быть инициирована данной активностью (последствие).
<i>includes</i>	Деятельность, на которой основана текущая активность.
	Слоты без перечисленных выше классификаторов содержат объекты, которые подвергаются воздействию или являются участниками деятельности.

Необходимо отметить, что слоты, помеченные классификатором *extends*, всегда располагаются в первой секции фрейма (т.е. в самом начале списка); за ними могут следовать *includes*-слоты, а уже затем все остальные. Исключением являются *activity*-фреймы – для них порядок следующий: входные параметры, выходные параметры, *includes*-слоты, прочие слоты. У экземпляров классов и прототипов первый слот должен иметь классификатор *class* или *prototype* и ссылаться на соответствующий объект.

Квалификаторы

Квалификатор – это предикат, записанный в форме *F-выражения*. В модели СФ квалификаторы служат для внесения ограничений, проверки соответствия условиям и т.п. Следует сразу указать, что система представления знаний с помощью СФ все еще находится на стадии разработки и апробации, поэтому в текущей спецификации потенциал *F-выражений* реализуется не полностью. На данный момент квалификаторы могут использоваться в качестве триггеров для проверки значений, присваиваемых слотам.

F-выражение (от англ. *F-expression* – functional expression) представляет собой запись, идентичную математической нотации определения *функции* [5], которая в форме *EBNF* [6] выглядит так:

```
f-expression = identifier, '(', arguments, ')';
arguments    = [ argument, { ',', argument } ];
argument     = value;
array        = '[', [ value, { ',', value } ], ']';
value        = f-expression | identifier | constant | array.
```

Причины, по которым был выбран такой синтаксис:

- 1) *F-выражения* по своей природе подобны функциям и не должны иметь побочных эффектов;
- 2) данный синтаксис, в отличие от *M-выражений* [7], используется для объявления/вызова функции во многих языках программирования (ЯП), производных от Algol, и, поэтому, является более привычным для восприятия;
- 3) данный синтаксис, в отличие от *символических выражений* [8], не привязан к конкретному ЯП (Lisp) и, используя *регулярные выражения* [9], может быть легко сконвертирован в AST [10], построенное на базе простых списков.

В отличие от обычных функций, *F-выражения* получают свои параметры в исходной форме и могут вычислять их по мере необходимости. Это позволяет унифицировать понятия *макрос* и *функция*.

Так, в зависимости от значения первого аргумента, форма *if(..., ..., ...)* вычисляет и возвращает либо второй, либо третий параметр.

Каждый квалификатор, будучи предикатом, по сути, должен возвращать строгое *логическое значение* [11]: $P() \in \{ \perp, T \}$.

Следует заметить, что в концепции *F-выражений* используется подобие *троичной логики* [12], где значения $(-1, 0, 1)$ представлены соответственно константами **\$f**, **\$?** и **\$t**.

Результаты логических операций над ними представлены в таблице 8.

Таблица 8. – Логические операции в троичной логике

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \oplus B$	$A \equiv B$
\$f	\$f	\$t	\$f	\$f	\$t	\$f	\$t
\$f	\$?	\$t	\$f	\$?	\$t	\$?	\$?
\$f	\$t	\$t	\$f	\$t	\$t	\$t	\$f
\$?	\$f	\$?	\$f	\$?	\$?	\$?	\$?
\$?	\$?	\$?	\$?	\$?	\$?	\$?	\$?
\$?	\$t	\$?	\$?	\$t	\$t	\$?	\$?
\$t	\$f	\$f	\$f	\$t	\$f	\$t	\$f
\$t	\$?	\$f	\$?	\$t	\$?	\$?	\$?
\$t	\$t	\$f	\$t	\$t	\$t	\$f	\$t

Важным дополнением является то, что константа **\$?** (undefined) представляет собой *суперзначение*, символизирующее отсутствие конкретных знаний как таковых, поэтому она может фигурировать в выражениях, оперирующих различными типами, например:

$\text{sum}(1,2,3, \$?) = \$?; \text{union}([1, 2, 3], [4, 5, 6], \$?) = \$?.$

По-умолчанию, значения слотов инициализируются данной константой.

Спецификация СФ в новой редакции включает следующие F-выражения:

- 1) математические операции: *neg, add, sub, mul, div, mod, rem, pow, exp, log, root2, root3*;
- 2) операторы сравнения: *below, below_or_equals, above, above_or_equals, equals, is_positive, is_negative*;
- 3) логические конструкции: *and, or, xor, not*;
- 4) предикаты для проверки типов: *is_number, is_string, is_array, is_boolean, is_strict_boolean, is_undefined*;
- 5) операции над контейнерами: *array, size, nth, is_empty, has*;
- 6) операции над множествами: *union, intersection, difference, symmetric_difference, contains*;
- 7) условные конструкции: *if*.

Заключение

Модель СФ претендует на статус универсальной системы представления знаний, включающей в себя свойства и возможности других наиболее распространенных нотаций. Насколько данное решение окажется приемлемым для разработчиков программного обеспечения, получив необходимую поддержку и стимул к дальнейшему развитию, зависит от множества факторов, в частности:

- 1) всесторонней продуманности модели, гибкости ее архитектуры и способности к расширению;
- 2) наличия исчерпывающей документации;
- 3) возможности и заинтересованности в стандартизации;
- 4) наличия достаточного количества примеров успешного применения и внедрения.

Если проанализировать текущую ситуацию на рынке коммерческого и в секторе свободного программного обеспечения, то можно заметить, что преобладающим является стремление к in-house разработке. Каждый год появляются новые бесполезные ЯП, форматы и протоколы, дистрибутивы операционных систем, библиотеки и фрейворки, дублирующие функциональность уже существующих решений. Это происходит потому, что особо консервативно настроенные компании, подверженные синдрому НИН (Not Invented Here [13]) предпочитают разрабатывать самостоятельно не только бизнес-логику, но и все сопутствующие технологии, тем самым пытаясь снизить риски, повысить конкурентно-способность и удержать финансовый приток от дочерних и партнерских фирм, использующих их решения. Что касается open-source сообщества, то оно страдает от так называемого «holy wars», неспособности к принятию единогогласных решений и организованной разработке.

Таким образом, в условиях текущей ситуации на рынке коммерческого и в секторе свободного программного обеспечения маловероятно, что система представления знаний, основанная на СФ, получит должное внимание и тем более широкое распространение.

ЛИТЕРАТУРА

1. Artificial neural network [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Artificial_neural_network. – Date of access: 20.09.2017.
2. First-order logic [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/First-order_logic. – Date of access: 20.09.2017.
3. Semantic network [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Semantic_network. – Date of access: 20.09.2017.
4. Frame (artificial intelligence) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/Frame_\(artificial_intelligence\)](https://en.wikipedia.org/wiki/Frame_(artificial_intelligence)). – Date of access: 20.09.2017.
5. Function (mathematics) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/Function_\(mathematics\)](https://en.wikipedia.org/wiki/Function_(mathematics)). – Date of access: 24.09.2017.
6. Extended Backus–Naur form [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Extended_Backus–Naur_form. – Date of access: 24.09.2017.
7. M-expression [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/M-expression>. – Date of access: 24.09.2017.
8. S-expression [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/S-expression>. – Date of access: 24.09.2017.
9. Regular expression [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Regular_expression. – Date of access: 24.09.2017.
10. Abstract syntax tree [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Abstract_syntax_tree. – Date of access: 24.09.2017.
11. Boolean algebra [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Boolean_algebra. – Date of access: 25.09.2017.
12. Three-valued logic [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Three-valued_logic. – Date of access: 25.09.2017.
13. Not invented here [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Not_invented_here. – Date of access: 25.09.2017.

УДК 004.[822+823]

**СЕМАНТИЧЕСКИЕ ФРЕЙМЫ:
РАЗРАБОТКА ФОРМАТА СЕРИАЛИЗАЦИИ ДАННЫХ****А.И. ДУНЧЕНКО****(Представлено: Д.В. ПЯТКИН)**

Представлено понятие о семантических фреймах. Сделан вывод о необходимости разработки формата сериализации для этой модели представления знаний, перечисляется перечень проблем, которые при этом требуют решения. Выполнен анализ наиболее очевидных способов достижения цели. Показано, как существующие наработки и стандарты могут быть адаптированы для решения поставленной задачи.

Семантические фреймы (СФ) – это универсальная система представления знаний об окружающем нас мире средствами ЭВМ, которая позволяет интерпретировать и структурировать поступающую информацию в виде, приемлемом для понимания как человеком (как правило, экспертом в конкретной проблемной области), так и компьютерной системой, выполняющей обработку этой информации.

Семантические фреймы являются гибридной моделью, базирующейся на концепциях *семантических сетей* [1] и *фреймов* [2] (отсюда и соответствующее название) и дополняющей их элементами *логики первого порядка* [3] и *нечеткой логики* [4].

При проектировании любого программного обеспечения, независимо от его сложности и того, выполняет ли оно конвейерную обработку данных или глобально оперирует средой выполнения (имеет побочные эффекты), всегда возникает вопрос о необходимости введения фиксированных механизмов обмена данными – протоколов взаимодействия – между компонентами системы: функциями, объектами, модулями или даже отдельными приложениями [5]. Примерами такого взаимодействия являются: *FFI* [6], *CGI* [7], *FIFO* [8], *сокеты* [9], *каналы* [10], сообщения.

В связи с этим рассмотрению подлежат следующие проблемы:

1) *обеспечение аппаратной независимости*, которое столь же актуально и в настоящее время, когда большинство настольных пользовательских систем работает на little-endian i386- и amd64-платформах, мобильные устройства – на dual-endian ARM-процессорах, а в сети Internet стандартным считается big-endian порядок байтов;

2) *обеспечение независимости от типа операционной и файловой систем*, так как даже при использовании виртуальных машин и интерпретируемых высокоуровневых языков программирования, по-прежнему имеют место трудности в реализации, обусловленные кодировками, регистро-(не)зависимостью, механизмами работы с файлами, устройствами и т.п.;

3) *обеспечение независимости от применяемого языка программирования (ЯП) или версии компилятора* (наиболее ярко выражена эта проблема в open-source сообществе, где разработчики имеют негативную тенденцию к внесению изменений в ABI [11] или API [12] с выходом каждой новой мажорной версией своего ПО);

4) *in-house разработка или применение уже существующего стандарта* (последнее, как правило, является более предпочтительным решением).

Модель СФ представляет собой не только спецификацию, но и фактическую реализацию – библиотеку, написанную на ЯП Common Lisp, – которая предназначена для работы в составе более сложной интеллектуальной программной системы. Поэтому для нее также справедливы все перечисленные ранее проблемы, ведь конечное ПО в процессе своего функционирования предполагает:

- 1) сохранение и чтение файлов с жесткого диска;
- 2) обмен данными между компонентами посредством сокетов;
- 3) отображение информации пользователю.

Таким образом, необходимо разработать формат для сериализации данных, который позволит не только эффективно выполнять операции ввода/вывода, но и будет приемлемым для восприятия человеком (экспертом или программистом, выполняющим проверку/отладку системы). Следуя этим требованиям, которые, кстати, соответствуют Unix-философии, целесообразно применить текстовый формат. При этом саму проблему можно разделить на две подзадачи:

1) там, где предполагается оперирование данными только в рамках библиотеки – сохранение и чтение из файлов – использовать *символические выражения* [13], на которых, собственно, и базируется синтаксис Common Lisp;

2) в случае если необходимо передавать данные между программами посредством CGI или сокетов (например, между сервером базы знаний и веб-сервером), утилизировать один из существующих текстовых языков разметки [14], адаптировав его под конкретные нужды.

Среди всех существующих языков разметки и представления данных наиболее подходящими являются следующие кандидаты: JSON, YAML, XML.

Рассмотрим их более подробно.

JSON (JavaScript Object Notation [15]) является подмножеством синтаксиса языка JavaScript. Считается, что, несмотря на свое происхождение, JSON 1) не зависит от JavaScript и может использоваться с любым другим языком программирования и 2) за счет своей лаконичности по сравнению с XML, может быть более подходящим для сериализации сложных структур. Однако при детальном рассмотрении оказывается, что оба эти утверждения далеки от истины.

Во-первых, система типов JSON полностью соответствует таковой именно языка JavaScript, поэтому, в сравнении с большинством других ЯП, зачастую либо оказывается крайне примитивной, либо не позволяет выполнить однозначную конверсию между значениями. Примеры: 1) в Common Lisp нет различий между константами *null* и *false*, оба значения представлены как *nil*, в Scheme вместо *nil* вообще используется пустой список; 2) символы и комплексные числа (C, Common Lisp, Scheme, Ruby, Python) приходится представлять в виде строки, в результате чего выполнить обратную интерпретацию в ряде случаев невозможно (что есть "4" – строка, символ или число?); 3) объекты JSON не являются полноценными hash-таблицами – в отличие от, к примеру, C++, Ruby или Python, ключами могут быть только строки.

Во-вторых, простой по сравнению с тем же XML синтаксис в то же время является и недостатком, так как провоцирует избыточные вложенные структуры там, где в XML можно обойтись лишь атрибутами. Кроме того, JSON просто-напросто не умеет представлять рекурсивные структуры.

YAML (YAML Ain't Markup Language [16]) ставит перед собой следующие задачи:

1) быть выразительным, расширяемым и доступным в понимании человеку;
2) поддерживать структуры данных, характерные для различных ЯП, обеспечивая легкую переносимость;

3) быть простым в реализации и использовании за счет обработки в один проход.

YAML имеет больше типов данных, чем JSON, поддерживает теги и ссылки на блоки. И тем не менее этот формат также не желателен в качестве решения для поставленной задачи. Создатели формата, как и автор языка Python, повторили ту же ошибку, сделав пробельные символы синтаксическим элементом. Это часто является причиной трудно обнаруживаемых ошибок и приводит к значительным трудностям как при визуальном просмотре многостраничных текстов, так и при написании утилит, предназначенных для анализа (обработки) документов. Иначе говоря, YAML является приемлемым вариантом, если речь идет о работе с файлами конфигурации или небольшой базой данных, но для представления таких сложных структур, как фреймы, этот формат не подходит.

Что касается XML (eXtensible Markup Language [17]), то несмотря на предостаточное количество критики в адрес данного формата (например, что для обработки документов требуется больше оперативной памяти и вычислительных ресурсов, чем в случае с JSON/YAML), он является если не идеальным, то вполне обоснованным решением проблемы: гибкий синтаксис XML позволяет 1) адаптировать его под любые задачи, связанные с разметкой и 2) сериализовать сколь угодно сложные иерархические структуры данных. Таким образом, взамен потери производительности мы получаем необычайно мощное средство представления данных, уступающее по своим возможностям только S-выражениям.

Определившись с форматом можно приступить к этапу проектирования структуры документа, т. е. опираясь на иерархию классов, предназначенных для описания и работы с фреймами, определить перечень используемых тегов, их атрибутов, а также правила формирования документа. Для этого можно воспользоваться формальной системой для определения синтаксиса EBNF (extended Backus-Naur form [18]):

document	= prologue, pool node;
prologue	= '<?xml version="1.0" encoding="UTF-8" ?>';
pool node	= '<pool', [' name="", pool name, ""], '>', { frame node }, '</pool>';
frame node	= '<frame id="", frame id, ""', [' name="", frame name, ""], '>', frame head node, frame body node, '</frame>';
frame head node	= '<head>', [classifiers node], [clarifiers node], '</head>';
classifiers node	= '<classifiers>', { classifier node }, '</classifiers>';
classifier node	= '<c>', classifier, '</c>';
clarifiers node	= '<clarifiers>', { clarifier node }, '</clarifiers>';
clarifier node	= ' ', predicate, ' ';
frame body node	= '<body>', { slot node slot reference node }, '</body>';

slot node	= '<slot id=" ', slot id, ' " ', [' name = " ', slot name, ' " '], '>', slot head node, slot data node, '</slot>';
slot head node	= '<head>', [classifiers node], [clarifiers node], [membership node], '</head>';
membership node	= '<membership>', membership, '</membership>';
membership	= <i>real number between 0 and 1</i> ;
slot data node	= '<data type=" ', slot data type, ' ">', slot data, '</data>';
slot data type	= 'boolean' 'character' 'number' 'string' 'JSON' 'S-exp', 'regex', 'base16' 'base64' 'slot-ref' 'frame-ref' custom data type;
custom data type	= <i>string</i> ;
slot reference node	= '<slot-ref id=" ', slot id, ' " />';
frame id	= id;
slot id	= id;
id	= <i>string</i> .

Как видно из правил, если пул, фрейм или слот не имеет идентификатора, атрибут *name* следует опускать. Для классификаторов и квалификаторов, которые с большой долей вероятности будут самыми часто встречающимися элементами, используются краткие теги *<c>* и *<p>* (сокращения от *classifier* и *predicate* соответственно). Сами предикаты вписываются в виде исходных *F-выражений* (F-expressions), а не в полученной после парсинга символической форме. Если слот принадлежит фрейму, то используется элемент *<slot>*, иначе (например, если фрейм ссылается на слот с классификатором *shared*) – *<slot-ref>*.

Теперь, когда общая структура документа четко определена, следует акцентировать внимание на промежуточном представлении СФ с помощью символических выражений, т. е. выбрать такой способ организации данных, который позволит с легкостью не только генерировать XML-разметку, но и выполнять обратную операцию – строить фреймы на основе данной формы.

Эта задача легко решается, если для работы с XML использовать пакет *cl-xmls*, который, кстати, и использует S-выражения в качестве входных/выходных данных. Построение каждого элемента починено весьма простому правилу, которое в нотации EBNF выглядит следующим образом:

node	= '(', node name, node attributes, node children, ')';
node name	= <i>literal</i> ;
node attributes	= '(', { attribute }, ')';
attribute	= '(', attribute name, attribute value, ')';
attribute name	= <i>literal</i> ;
attribute value	= <i>value</i> ;
node children	= <i>value</i> { node };
literal	= <i>string</i> <i>symbol</i> ;
value	= <i>any Lisp object but pair</i> .

Такой незаурядный способ организации данных позволяет эффективно выполнять их обработку, применяя функциональный подход к программированию и встроенные в стандартную библиотеку языка Common Lisp макросы и функции. Более того, учитывая тот факт, что *cl-xmls* автоматически конвертирует любые значения в строки, можно существенно сэкономить память, если в качестве имен тегов использовать символы из пакета *keywords*.

Подводя итоги, сделаем вывод, что поставленная задача по разработке формата для сериализации СФ успешно решена: для внутренних нужд библиотеки, реализующей спецификацию СФ, будет использоваться промежуточное представление, основанное на символических выражениях, а для межпроцессного взаимодействия, где требуется обеспечить максимальную изолированность компонентов и независимость от программных решений, – язык разметки XML.

Единственной нерешенной проблемой остается сериализация триггеров. Причины, по которым это на данный момент представляется невозможным, следующие:

- 1) при сохранении замыканий нужно также сохранять информацию о среде выполнения, а в случае с Common Lisp, контекст может быть как статическим, так и динамическим;
- 2) триггеры могут быть написаны на разных ЯП, не обязательно Common Lisp;

3) стек может и вовсе содержать цепочку вызовов из разных ЯП, например:
 $C \rightarrow \text{Common Lisp} \rightarrow C \rightarrow \text{Tcl} \rightarrow C \rightarrow \text{Common Lisp} \rightarrow C$,
в результате чего выполнить полный захват и последующую корректную инициализацию невозможно в принципе.

ЛИТЕРАТУРА

1. Semantic network [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Semantic_network. – Date of access: 20.09.2017.
2. Frame (artificial intelligence) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/Frame_\(artificial_intelligence\)](https://en.wikipedia.org/wiki/Frame_(artificial_intelligence)). – Date of access: 20.09.2017.
3. First-order logic [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/First-order_logic. – Date of access: 20.09.2017.
4. Fuzzy logic [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Fuzzy_logic. – Date of access: 20.09.2017.
5. Inter-process communication [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Inter-process_communication. – Date of access: 20.09.2017.
6. Foreign function interface [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Foreign_function_interface. – Date of access: 20.09.2017.
7. Common Gateway Interface [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Common_Gateway_Interface. – Date of access: 20.09.2017.
8. FIFO (computing and electronics) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/FIFO_\(computing_and_electronics\)](https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics)). – Date of access: 20.09.2017.
9. Unix domain socket [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Unix_domain_socket. – Date of access: 20.09.2017.
10. Pipeline (computing) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/Pipeline_\(computing\)](https://en.wikipedia.org/wiki/Pipeline_(computing)). – Date of access: 20.09.2017.
11. Application binary interface [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Application_binary_interface. – Date of access: 20.09.2017.
12. Application programming interface [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Application_programming_interface. – Date of access: 20.09.2017.
13. S-expression [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/S-expression>. – Date of access: 21.09.2017.
14. Markup language [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Markup_language. – Date of access: 21.09.2017.
15. JSON [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/JSON>. – Date of access: 21.09.2017.
16. YAML [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/YAML>. – Date of access: 21.09.2017.
17. XML [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/XML>. – Date of access: 21.09.2017.
18. Extended Backus–Naur form [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Extended_Backus%E2%80%93Naur_form. – Date of access: 21.09.2017.

УДК 004.65

**ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ
ВЕБ-ПРИЛОЖЕНИЯ «АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА
ВЕДЕНИЯ ОТЧЕТНОСТИ ДЛЯ ОАО «НАФТАН»»****А.Н. РОДЧЕНКОВ***(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)*

Представлен объект разработки – веб-приложение «Автоматизированная информационная система ведения отчетности для ОАО «Нафтан»». Рассмотрены вопросы проектирования базы данных для информационной системы ведения отчетности.

Цель работы – создание веб-приложения «Автоматизированная информационная система ведения отчетности для ОАО «Нафтан»». Для создания приложения, работающего с базой данных, необходимо сначала спроектировать саму базу. Для удобства последующей разработки проектируемая база должна быть максимально проста для понимания и нормализована. Для проектирования базы данных необходимо выявить необходимый набор сущностей, указать необходимые атрибуты, характеризующие каждую сущность, а также определить связи между сущностями.

На предприятии имеется в наличии различное измерительное оборудование, которое подлежит складскому учету. При учете имеющегося оборудования указывается заглавие секции, на балансе которой оно находится. Оборудование делится на два типа: клапаны и приборы. Исходя из этого получаем следующий набор сущностей:

- 1 «Клапаны».
- 2 «Приборы».
- 3 «Секции».
- 4 «Склад».

Связь между сущностями «Клапаны», «Приборы», «Секции» и «Склад» представлена на рисунке 1.

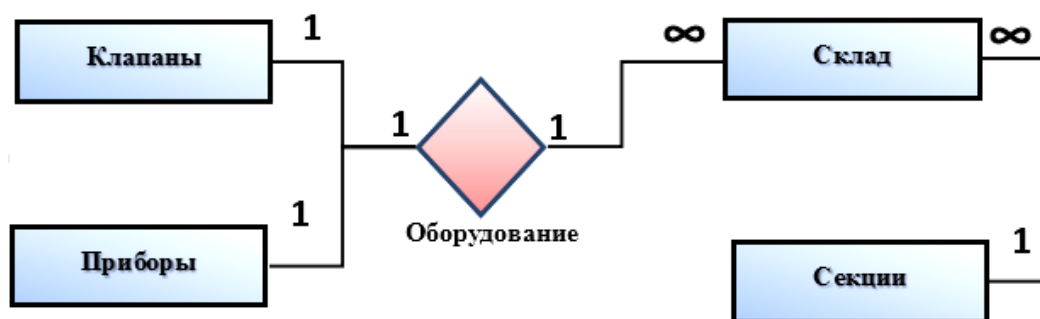


Рисунок 1. – Связь между сущностями «Клапаны», «Приборы», «Секции» и «Склад»

Также имеются различные технические позиции, за которыми, в свою очередь, закрепляются позиции размещения измерительных клапанов. На данных позициях размещается различное измерительное оборудование, которое есть на складе. Для фиксации закреплённого складского оборудования на позициях необходимо определить такую сущность, как «Закрепление». Для более детального представления о закреплении различного рода оборудования на технических позиция и позициях измерительных клапанов введем дополнительные сущности – «Закрепление приборов» и «Закрепление клапанов».

Исходя из набора вышеописанных фактов список выявленных сущностей будет выглядеть следующим образом:

- 1 «Тех. Позиция».
- 2 «Позиция ИКС».
- 3 «Закрепление Клапанов».
- 4 «Закрепление Приборов».

Связь между сущностями «Тех. Позиция», «Позиция ИКС», «Закрепление Клапанов» и «Закрепление Приборов» представлена на рисунке 2.

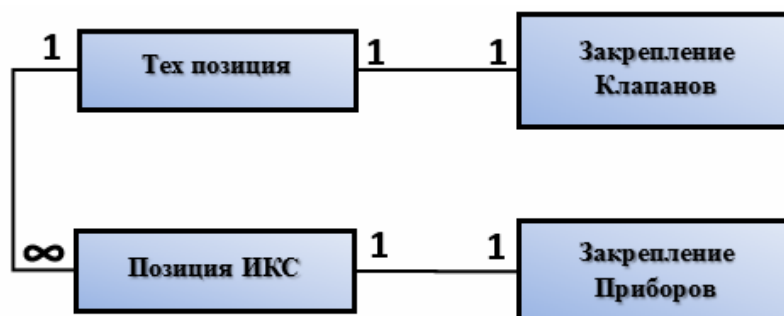


Рисунок 2. – Связь между сущностями «Тех. Позиция», «Позиция ИКС», «Закрепление Клапанов» и «Закрепление Приборов»

Для построения схемы базы данных необходимо определить совокупность отношений между сущностями, которые будут использоваться в базе. Также важно, чтобы в отношениях между сущностями выполнялись условие целостности сущностей и условие целостности ссылок.

Условие целостности сущностей заключается в следующем: каждый кортеж любого отношения должен отличаться от любого другого кортежа этого отношения, то есть любое отношение должно обладать первичным ключом, а условие целостности ссылок, в свою очередь, заключается в том, что для каждого значения внешнего ключа, появляющегося в дочернем отношении, в родительском отношении должен найтись кортеж с таким же значением первичного ключа [3].

Схема спроектированной базы данных представлена на рисунке 3.

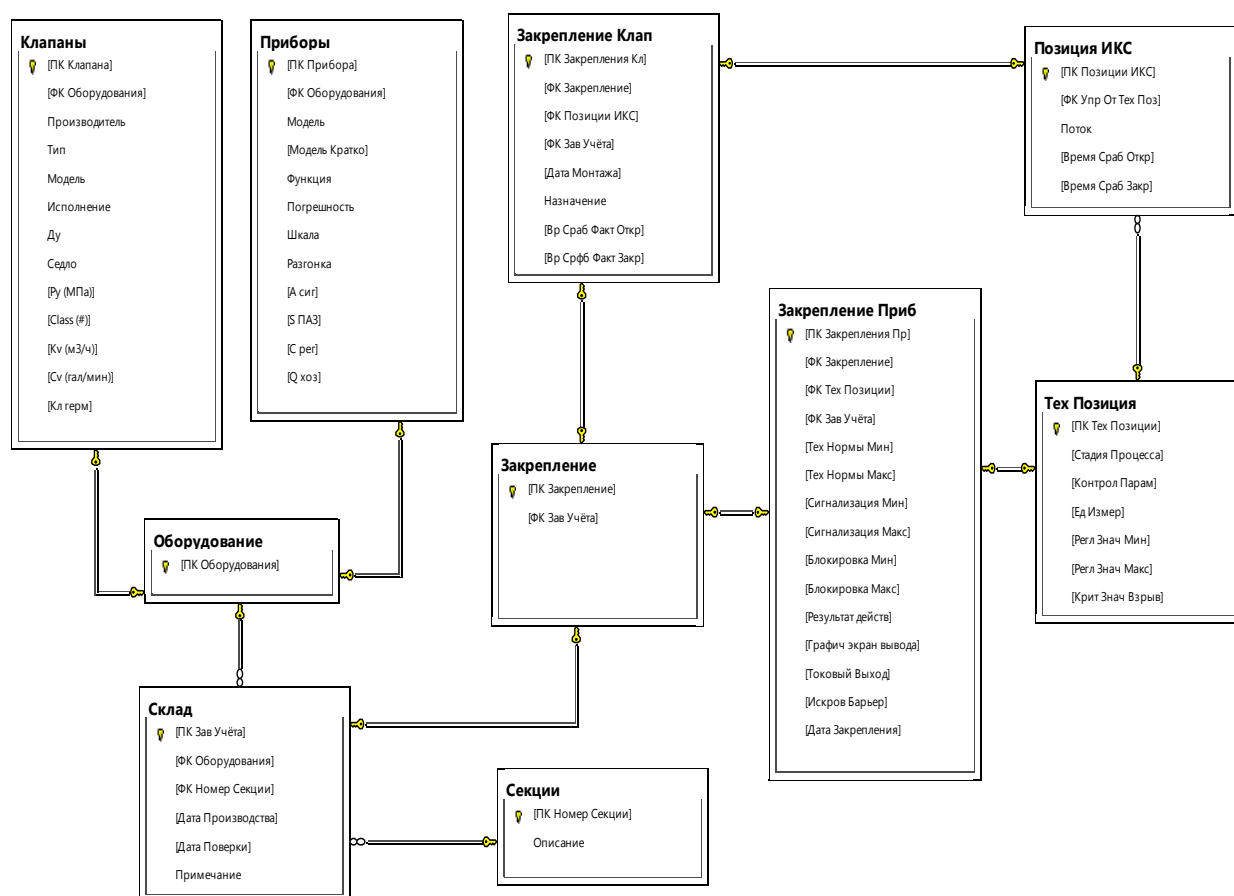


Рисунок 3. – Набор необходимых отношений базы данных

Выбор СУБД является сложной задачей и должен основываться, в первую очередь, на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид

программного продукта и категория пользователей (или профессиональные программисты, или конечные пользователи, или и то, и другое).

Другими показателями, влияющими на выбор СУБД, являются [2]:

- удобство и простота использования;
- качество средств разработки, защиты и контроля базы данных;
- уровень коммуникационных средств в случае применения ее в сетях;
- фирма-разработчик;
- стоимость.

Среди всего множества доступных СУБД выбор был сделан в пользу MS SQL Server 2014. Выбор средств решения поставленной задачи исходил из того, какое программное и аппаратное обеспечение используется в данный момент на предприятии.

Система MS SQL Server 2014 позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio. MS SQL Server обеспечивает высочайший уровень безопасности, надежности и масштабируемости для критически важных приложений. Чтобы использовать новые возможности, постоянно возникающие в быстро меняющемся деловом мире, предприятиям нужно быть способными быстро создавать и развертывать решения, управляемые данными. MS SQL Server 2014 позволяет сократить затраты времени и средства, требуемые на управление и развертывание таких приложений. Следует учесть, что фирма-разработчик данной СУБД является также разработчиком самой распространенной ОС. В финансовом плане важным фактором является то, что существуют бесплатные сборки данной СУБД.

Для правильной работы и сохранения ссылочной целостности для сущностей в базе были разработаны триггеры на добавление/удаление/редактирование, которые предотвращают добавление в базу одинаковых записей и организуют каскадное удаление, не оставляя неактуальные ссылки.

Заключение

В результате проделанной работы спроектирована база данных, при помощи которой можно создать автоматизированную информационную систему, направленную на сокращение временных затрат работников предприятия, занимающегося ведением различной внутренней документации и автоматизировать формирование отчетов.

ЛИТЕРАТУРА

1. Малыгина, М. Базы данных: основы, проектирование, использование / М. Малыгина. – М. : БНВ, 2004. – 512 с.
2. Статья «Выбор СУБД» [Электронный ресурс]. – Режим доступа <http://websideworld.com/main/data-storage/6/2>. – Дата доступа: 27.09.2017.
3. Зависимости по соединению и пятая НФ (5NF). [Электронный ресурс]. – Режим доступа: <http://sdamzavas.net/3-6246.html>. – Дата доступа: 27.09.2017.

УДК 004.031.042

**ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНОЙ ЧАСТИ
ВЕБ-ПРИЛОЖЕНИЯ «АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА
ВЕДЕНИЯ ОТЧЕТНОСТИ ДЛЯ ОАО «НАФТАН»»**

А.Н. РОДЧЕНКОВ*(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)*

Представлен объект разработки – веб-приложение «Автоматизированная информационная система ведения отчетности для ОАО «Нафтан»». Рассмотрены вопросы проектирования функциональной части веб-приложения «Автоматизированная информационная система ведения отчетности для ОАО «Нафтан»».

Создание программного обеспечения для надежного хранения данных о функционировании предприятия и формирование внутренних отчетов является целью данного исследования. Основная цель разработки приложения – сделать хранение данных о функционировании предприятия и формирование отчетов как можно больше простым и универсальным.

Первым этапом проектирования программного обеспечения является *определение функциональных и других возможностей, которым программное обеспечение должно соответствовать*. Предполагается, что с данной информационной системой будут работать сотрудники установки, и они должны иметь возможность решать с ее помощью следующих задач:

- ввод и редактирование данных о технологических позициях, приборах и закреплениях в удобной форме;
- просмотр данных о технологических позициях, приборах и закреплениях в отсортированном виде;
- просмотр и подготовка к выводу списков технологических позиций и закрепленных за ними приборов;
- формирование отчетов согласно стандартам технического регламента;
- экспорт информации и сформированных отчетов из базы данных в Excel и наоборот;
- организация надежного хранения данных;
- организация поиска по ID или части ID.

Таким образом, при разработке продукта были выдвинуты следующие требования к функциональным характеристикам:

- предоставлять пользователю информацию, хранящуюся в информационной системе;
- предоставлять базовый функционал сортировки информации при работе с информационной системой;
- предоставлять пользователю возможность вносить изменения в информацию, хранящуюся в информационной системе;
- предоставлять пользователю возможность экспорта информации из базы данных в Excel и наоборот;
- предоставлять авторизованному пользователю возможность формирования отчетов согласно внутреннему техническому регламенту.

Так как разрабатываемое программное средство является автоматизированной информационной системой, то требования, предъявляемые к любой автоматизированной информационной системе, будут распространены и на разрабатываемый программный продукт. Среди них выделяют:

1) наличие визуальной среды для просмотра и обработки информации. Для обработки информации, которая поступает в информационную систему, необходимо разработать удобную оболочку, через которую, посылая запросы к модулям логики работы системы, пользователь осуществляет необходимую ему обработку данных. Для такой визуальной среды важными характеристиками являются дизайн пользовательского интерфейса, интуитивность и удобство навигации;

2) возможность выполнять с информацией, поступающей в информационную систему, различные стандартные операции. К таким операциям относят операции добавления, удаления, а также изменения (редактирования) данных;

3) логическое разделение автоматизированной информационной системы на программные компоненты. В данном случае информационную систему удобнее всего разделить на четыре основных вышеперечисленных компонента: база данных, подсистема генерации отчетов, подсистема импорта данных из Excel, редактор базы данных;

4) ориентированность на пользователя – для работы с разрабатываемым программным средством не требуется предварительной подготовки персонала;

5) стабильность работы – приложение должно обеспечивать возможность корректного завершения в случае ошибки.

Следующим этапом проектирования является *разработка функциональной структуры программного обеспечения*. В данном случае информационную систему удобнее всего разделить на четыре основных компонента: база данных, подсистема генерации отчетов, подсистема импорта данных из Excel, редактор базы данных.

Среди функций разрабатываемых подсистем можно выделить следующее:

- 1 Для подсистемы база данных:
 - удобное и надежное хранение необходимой информации;
 - обеспечение целостности хранимой информации.
- 2 Для подсистемы импорта данных из Excel:
 - непосредственно импорт данных из Excel в базу данных;
 - возможность дополнения или обновления имеющийся базы с помощью импорта данных из Excel;
 - возможность выбора заполняемой таблицы;
 - возможность выбора файла для импорта данных из Excel.
- 3 Для подсистемы генерации отчетов:
 - генерация отчетов различных типов;
 - возможность просмотра генерируемых отчетов;
 - изменение параметров генерации отчетов;
 - возможность сохранения отчетов в различных форматах.
- 4 Для подсистемы редактор базы данных:
 - добавление записей в базу данных согласно формату;
 - удаление записей из базы данных;
 - изменение записей в базе данных по ключу;
 - различные варианты группировок при работе с базой данных для более удобной обработки информации;
 - возможность ускоренного поиска по части слова в выбранной ячейке;
 - возможность скрыть меню, расширение рабочей области;
 - ограничение количества выводимых полей на одной странице.

Следующим этапом проектирования программного обеспечения является *создание диаграммы вариантов использования* [1].

При создании диаграмм вариантов использования нужно выделить два вида сущностей – актеры и прецеденты. Актер – это множество логически связанных ролей, исполняемых при взаимодействии с прецедентами. Прецедент – это описание множества последовательных событий, включая варианты, выполняемые системой, которые приводят к наблюдаемому актером результату.

На диаграмме вариантов использования указан один актер – «Сотрудник», который совершает все действия, выделенные при проектировании. Диаграмма вариантов использования представлена на рисунке.

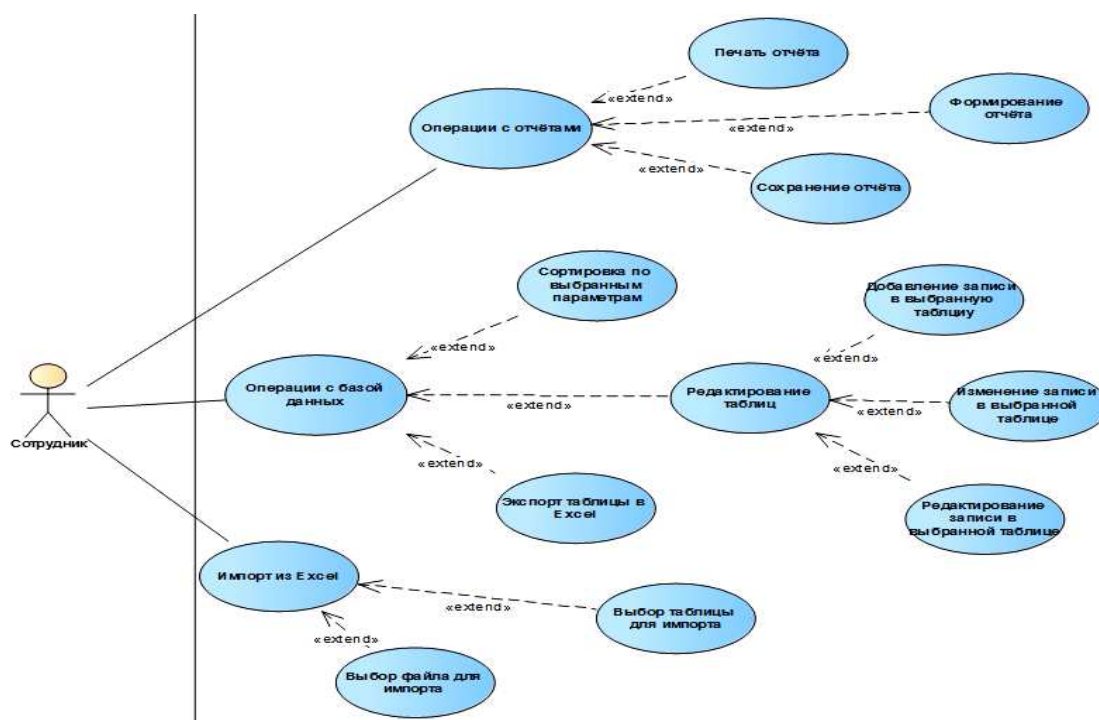


Диаграмма вариантов использования

Последний этап проектирования функциональной части программного обеспечения – *выбор инструментов*.

Выбор средств решения поставленной задачи исходил из того, какое программное и аппаратное обеспечение используется в данный момент на предприятии.

Выбор СУБД – сложная задача, должен основываться, в первую очередь, на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид программного продукта и категория пользователей. Другими показателями, влияющими на выбор СУБД, служат [2]:

- удобство и простота использования;
- качество средств разработки, защиты и контроля базы данных;
- уровень коммуникационных средств в случае применения ее в сетях;
- фирма-разработчик;
- стоимость.

Система MS SQL Server 2014 позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio. MS SQL Server обеспечивает высочайший уровень безопасности, надежности и масштабируемости для критически важных приложений. Чтобы использовать новые возможности, постоянно возникающие в быстро меняющемся деловом мире, предприятиям нужно быть способными быстро создавать и развертывать решения, управляемые данными. MS SQL Server 2014 позволяет сократить затраты времени и средств, требуемые на управление и развертывание таких приложений. Также следует учесть, что фирма-разработчик данной СУБД является также разработчиком самой распространенной ОС. В финансовом плане важным фактором является то, что существуют бесплатные сборки данной СУБД (Express).

Для реализации приложения была выбрана среда разработки Microsoft Visual Studio 2013, в качестве языка программирования – C#.

Достоинства платформы .NET [3]:

1 Вся платформа .NET основана на единой объектно-ориентированной модели. Все сервисы, интерфейсы и объекты, которые платформа предоставляет разработчику, объединены в единую иерархию классов. Другими словами, все, что может потребоваться при создании приложений под платформу .NET будет всегда у вас под рукой. Причем все это сгруппировано удобно и интуитивно понятно.

2 Приложение, написанное на любом .NET-совместимом языке, является межплатформенным (в идеале). Почему в идеале? Дело в том, что приложение, написанное, скажем, на том же C#, не зависит от платформы, на которой будет выполняться, но зато зависит от наличия платформы .NET.

3 В состав платформы .NET входит «сборщик мусора», который освобождает ресурсы. Таким образом, приложения защищены от утечки памяти и от необходимости освобождать ресурсы. Это делает программирование более легким и более безопасным.

4 Приложения .NET используют безопасные типы, что повышает их надежность и совместимость.

5 NET приложения могут быть сертифицированы на безопасность. Это является особенностью промежуточного кода, в который преобразуются все .NET приложения.

6 Абсолютно все ошибки обрабатываются механизмом исключительных ситуаций. Это позволяет избежать разногласий, которые иногда возникают при программировании под Win32.

7 Повторное использование кода стало еще удобнее. Это связано с тем, что промежуточный язык MSIL не зависит от языка программирования. Например, вы можете написать программу на C#, а патч к ней писать уже, скажем, на J#.

Заключение

В результате проведенной работы спроектировано веб-приложение, позволяющее хранить, изменять, добавлять информацию в базу данных, осуществлять быстрый поиск и различную группировку информации с последующим формированием различных отчетов.

ЛИТЕРАТУРА

1. Лекция 3: Элементы графической нотации диаграммы вариантов использования [Электронный ресурс]. – Режим доступа: <http://www.intuit.ru/studies/courses/32/32/lecture/1004?page=2>. – Дата доступа: 27.09.2017.
2. Выбор СУБД [Электронный ресурс]. – Режим доступа <http://websideworld.com/main/data-storage/6/2>. – Дата доступа: 27.09.2017.
3. Нгуен, П. Достоинства и недостатки .Net / Павел Нгуен [Электронный ресурс]. – Режим доступа: http://blog.nguen.net/post33-good_bad_plus_minus_dot_net.html. – Дата доступа: 27.09.2017.

УДК 004.021

**КОМБИНИРОВАННЫЕ АЛГОРИТМЫ ШИФРОВАНИЯ,
ЦЕЛЕСООБРАЗНОСТЬ ИХ ИСПОЛЬЗОВАНИЯ ДЛЯ ЗАЩИТЫ ФАЙЛОВ
ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА****А.Г. АНДРЕЙЧИКОВ****(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)**

Рассматриваются возможности комбинированных алгоритмов шифрования. Показаны преимущества и недостатки комбинированных алгоритмов при их использовании для защиты файлов от несанкционированного доступа.

Комбинированный (гибридный) метод шифрования позволяет сочетать преимущества высокой секретности, предоставляемые асимметричными криптосистемами с открытым ключом, с преимуществами высокой скорости работы, присущими симметричным криптосистемам с секретным ключом. При таком подходе криптосистема с открытым ключом применяется для шифрования, передачи и последующего дешифровки только секретного ключа симметричной криптосистемы. А симметричная криптосистема применяется для шифрования и передачи исходного открытого текста. В результате криптосистема с открытым ключом не заменяет симметричную криптосистему с секретным ключом, а лишь дополняет ее, позволяя повысить в целом защищенность передаваемой информации.

Для эффективного использования гибридных криптосистем необходимо рассмотреть отдельно симметричные и асимметричные алгоритмы шифрования их преимущества и недостатки.

Симметричные алгоритмы – это алгоритмы шифрования, в которых для шифрования и дешифровки используется один и тот же ключ, что, в свою очередь, обеспечивает высокую скорость работы.

Алгоритмы шифрования данных широко применяются в компьютерной технике в системах сокрытия конфиденциальной и коммерческой информации от злонамеренного использования сторонними лицами. Главным принципом в них является условие, что передатчик и приемник заранее знают алгоритм шифрования, а также ключ к сообщению, без которых информация представляет собой всего лишь набор символов, не имеющих смысла.

Классическими примерами таких алгоритмов являются симметричные криптографические алгоритмы:

- Простая перестановка.
- Одиночная перестановка по ключу.
- Двойная перестановка.
- Перестановка «Магический квадрат».

Основными достоинствами данного типа алгоритмов является скорость и простота реализации. Также стоит отметить, что при одинаковой длине ключа криптостойкость симметричных алгоритмов в несколько раз выше, чем асимметричных. Данный вид алгоритмов шифрования появился достаточно давно, и хорошо изучен.

Так как для шифрования и дешифровки используется один и тот же ключ, то самой большой проблемой при использовании данного типа алгоритмов является передача данного ключа. Передача ключа должна осуществляться по секретным каналам обеим сторонам. Эта проблема порождает ещё один важный недостаток – сложность управления ключами в случае большой сети, что сильно усложняет обмен данными и контроль за секретностью ключей.

Асимметричные криптографические системы были разработаны в 1970-х годах. Принципиальное отличие асимметричной криптосистемы от криптосистемы симметричного шифрования состоит в том, что для шифрования информации и ее последующего расшифровывания используются различные ключи:

- открытый ключ K используется для шифрования информации, вычисляется из секретного ключа k ;
- секретный ключ k используется для расшифровывания информации, зашифрованной с помощью парного ему открытого ключа K .

Эти ключи различаются таким образом, что с помощью вычислений нельзя вывести секретный ключ k из открытого ключа K . Поэтому открытый ключ K может свободно передаваться по каналам связи.

Асимметричные системы называют также двухключевыми криптографическими системами, или криптосистемами с открытым ключом.

По сравнению с симметричными алгоритмами асимметричные имеют ряд преимуществ, однако есть и недостатки. Главное преимущество состоит в том, что нет необходимости предварительно передавать секретный ключ. Также преимуществом асимметричных алгоритмов перед симметричными является тот факт, что при использовании системы в большой сети количество ключей значительно меньше, что, в свою очередь, упрощает их распространение. В качестве главного недостатка выделим сложность

внесения изменений в алгоритм, поскольку при использовании алгоритма в практических целях часто возникает необходимость изменения алгоритма под конкретную задачу или проблему. Также нужно обратить внимание на большую размерность ключей по сравнению с симметричными алгоритмами при одинаковой криптостойкости.

В качестве примера работы симметричного алгоритма реализована простая программа для шифрования введенной строки.

Листинг 1 – Пример алгоритма шифрования 3DES

```
public static string Encrypt(string source, string key)
{
    TripleDESCryptoServiceProvider desCryptoProvider = new TripleDESCryptoServiceProvider();
    MD5CryptoServiceProvider hashMD5Provider = new MD5CryptoServiceProvider();

    byte[] byteHash;
    byte[] byteBuff;

    byteHash = hashMD5Provider.ComputeHash(Encoding.UTF8.GetBytes(key));
    desCryptoProvider.Key = byteHash;
    desCryptoProvider.Mode = CipherMode.ECB; //CBC, CFB
    byteBuff = Encoding.UTF8.GetBytes(source);

    string encoded =
    Convert.ToBase64String(desCryptoProvider.CreateEncryptor().TransformFinalBlock(byteBuff, 0, byteBuff.Length));
    return encoded;
}

public static string Decrypt(string encodedText, string key)
{
    TripleDESCryptoServiceProvider desCryptoProvider = new TripleDESCryptoServiceProvider();
    MD5CryptoServiceProvider hashMD5Provider = new MD5CryptoServiceProvider();

    byte[] byteHash;
    byte[] byteBuff;

    byteHash = hashMD5Provider.ComputeHash(Encoding.UTF8.GetBytes(key));
    desCryptoProvider.Key = byteHash;
    desCryptoProvider.Mode = CipherMode.ECB; //CBC, CFB
    byteBuff = Convert.FromBase64String(encodedText);

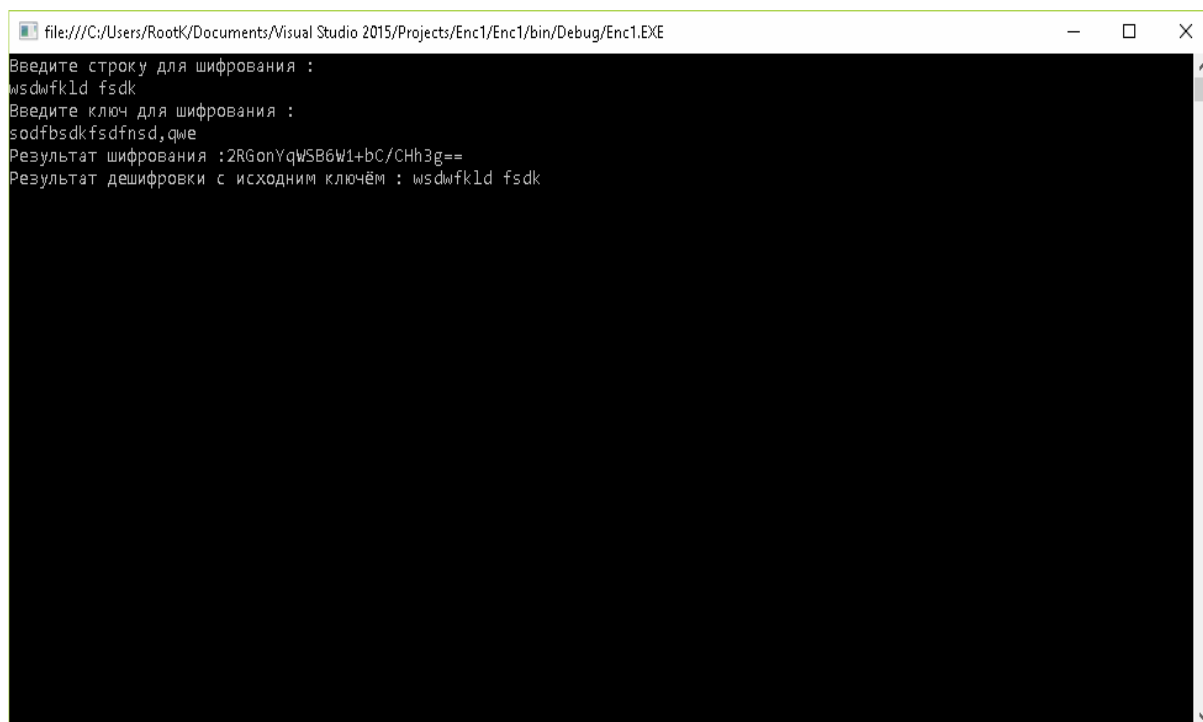
    string plaintext = Encoding.UTF8.GetString(desCryptoProvider.CreateDecryptor().TransformFinalBlock(byteBuff, 0, byteBuff.Length));
    byte[] byt = desCryptoProvider.CreateDecryptor().TransformFinalBlock(byteBuff, 0, byteBuff.Length);

    return plaintext;
}

static void Main(string[] args)
{
    Console.WriteLine("Введите строку для шифрования : ");
    String str = Console.ReadLine();
    Console.WriteLine("Введите ключ для шифрования : ");
    String key = Console.ReadLine();

    String result = Encrypt(str, key);
    Console.WriteLine("Результат шифрования : " + result);
    String deresult = Decrypt(result, key);
    Console.WriteLine("Результат дешифровки с исходным ключом : " + deresult);
    Console.ReadKey();
}
```

Результат работы алгоритма 3DES представлены на рисунке 1.



```
file:///C:/Users/RootK/Documents/Visual Studio 2015/Projects/Enc1/Enc1/bin/Debug/Enc1.EXE
Введите строку для шифрования :
wsdwfkld fsdk
Введите ключ для шифрования :
sodfbsdk fsdfnsd,qwe
Результат шифрования : 2RGonYqWSB6W1+bC/CHh3g==
Результат дешифровки с исходным ключём : wsdwfkld fsdk
```

Рисунок 1. – Результат работы алгоритма 3DES

Заключение

В данной статье представлен краткий обзор особенностей симметричных и асимметричных алгоритмов шифрования. Показаны преимущества и недостатки этих типов алгоритмов шифрования. В качестве примера симметричного типа шифрования представлен алгоритм 3DES и результаты работы простейшей программы с его использованием.

ЛИТЕРАТУРА

1. Хабрахабр [Электронный ресурс] Пара слов о гибридном шифровании и эллиптических кривых. – Режим доступа: <https://habrahabr.ru/post/106057>. – Дата доступа: 24.09.2017.
2. Хабрахабр [Электронный ресурс] Немного практической криптографии под .NET для чайников. – Режим доступа: <https://habrahabr.ru/post/254909>. – Дата доступа: 24.09.2017.
3. Хабрахабр [Электронный ресурс] RSA, а так ли все просто. – Режим доступа: <https://habrahabr.ru/post/99376>. – Дата доступа: 24.09.2017.

УДК 004.021

ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА ДЛЯ ЗАЩИТЫ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА К ФАЙЛАМ

А.Г. АНДРЕЙЧИКОВ*(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)*

Представлен практический способ создания приложения с использованием криптосистемы, основанной на гибридных алгоритмах шифрования. Результатом работы явилось написание программы для защиты пользовательских файлов с использованием гибридных алгоритмов шифрования, а также создание понятного пользовательского интерфейса. Задача реализации интерфейса решалась с использованием WindowsForm и языка программирования C#.

В настоящий момент самым распространенным способом хранения информации является запись ее на цифровые носители. Часть сохраняемой информации может быть конфиденциальной, приватной, секретной и нуждаться в защите. У такой информации есть законные пользователи. Но всегда существует вероятность появления пользователей незаконных, стремящихся захватить секретную информацию с целью обращения ее себе во благо. Хакеры ежедневно крадут номера кредитных карт, банковские счета и даже личность человека. Опасаясь этого, законные пользователи принимают меры по защите информации.

На сегодняшний день известны несколько подходов к проблеме защиты информации, хранящейся в персональном компьютере. Один из них – шифрование данных.

Принцип разработанного алгоритма защиты информации и его функционал. В качестве симметричного алгоритма шифрования использован алгоритм Triple DES (3DES). Triple DES (3DES) – симметричный блочный шифр, созданный Уитфилдом Диффи, Мартином Хеллманом и Уолтом Тачманном в 1978 году на основе алгоритма DES с целью устранения главного недостатка последнего – малой длины ключа (56 бит), который может быть взломан методом полного перебора ключа. Скорость работы 3DES в 3 раза ниже, чем у DES, но криптостойкость намного выше. 3DES является простым способом устранения недостатков DES. Алгоритм 3DES построен на основе DES, поэтому для его реализации можно использовать программы, созданные для DES.

В качестве асимметричного алгоритма шифрования выбран алгоритм RSA, который относится к так называемым асимметричным алгоритмам, у которых ключ шифрования не совпадает с ключом дешифровки. Один из ключей доступен всем и называется открытым ключом, другой хранится только у его хозяина и неизвестен никому другому. С помощью одного ключа можно производить операции только в одну сторону. Если сообщение зашифровано с помощью одного ключа, то расшифровать его можно только с помощью другого. Имея один из ключей, невозможно найти другой ключ, если разрядность ключа высока.

Для аутентификации пользователя используется система логин-пароль. В систему можно зайти с любой комбинацией логина-пароля, однако файлы, зашифрованные с помощью одной комбинации логина-пароля, невозможно дешифровать с помощью другой комбинации, даже введя верный ключ. Данный подход обеспечивает дополнительный уровень защиты от несанкционированного доступа к секретным данным.

Интерфейс программы создан с помощью WindowsForm на языке программирования C# (рис. 1).

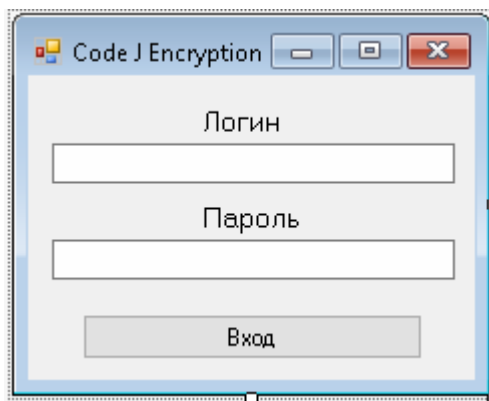


Рисунок 1. – Форма для авторизации в программе

После авторизации в программе и выбора действия интерфейс будет выглядеть, как показано на рисунках 2, 3.

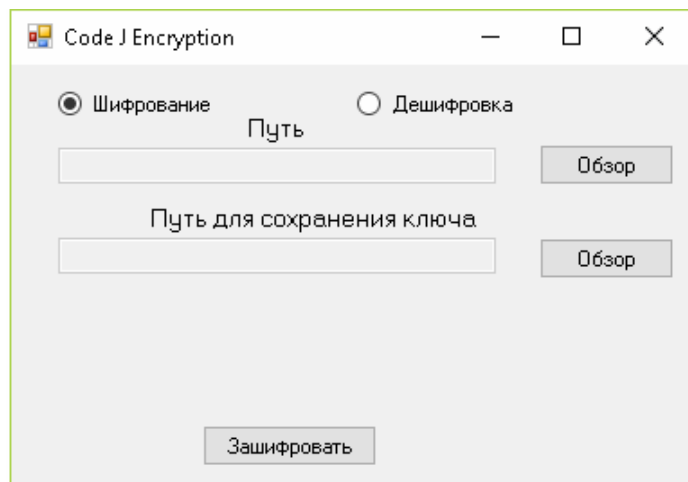


Рисунок 2. – Интерфейс шифрования

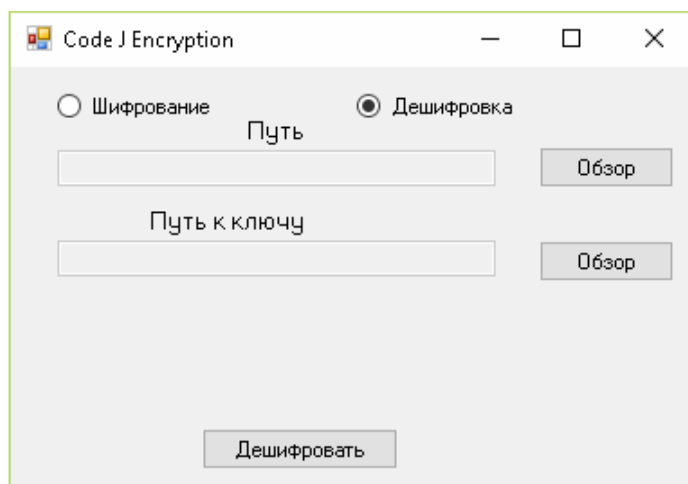


Рисунок 2. – Интерфейс дешифровки

Заключение

Автором разработан программный продукт для надежного хранения конфиденциальной, секретной информации на персональном компьютере. Ведутся работы по увеличению функционала программы и улучшению степени защиты данных.

ЛИТЕРАТУРА

1. Triple DES [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Triple_DES. – Дата доступа: 24.09.2017.
2. Хабрахабр [Электронный ресурс] Пара слов о гибридном шифровании и эллиптических кривых. – Режим доступа: <https://habrahabr.ru/post/106057>. – Дата доступа: 24.09.2017.
3. Хабрахабр [Электронный ресурс] Немного практической криптографии под .NET для чайников. – Режим доступа: <https://habrahabr.ru/post/254909>. – Дата доступа: 24.09.2017.
4. Хабрахабр [Электронный ресурс] RSA, а так ли все просто. – Режим доступа: <https://habrahabr.ru/post/99376>. – Дата доступа: 24.09.2017.

УДК 004.021

**ПРОЕКТИРОВАНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА
ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ «МУЗЫКАНТЫ РОССИИ»
ПОД ОПЕРАЦИОННУЮ СИСТЕМУ iOS**

А.В. ВЕЧЕРОВ*(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются принципы построения графического интерфейса пользователя в мобильном приложении под операционную систему iOS. Показаны их плюсы и минусы. Представлены примеры работы с каждой из систем.

В каждой операционной системе свои принципы построения графического интерфейса. В iOS для этого используются XML-файлы – Storyboards, либо программное создание разметки посредством кода. Расположение объектов формируется через задание рамки, координат и размера элементов интерфейса либо с помощью системы AutoLayout-ов, которые задают взаимные отступы.

Данная работа направлена на подробное описание технологий построения разметки в iOS.

Как правило, приложения в iOS строятся по шаблону MVC (Model-View-Controller или Модель-Вид-Контроллер). Идея данного шаблона проста – разделение обязанностей: задача контроллера – обработка действий пользователя (нажатия по кнопкам, обработка запросов к серверу и т.д.); модель предоставляет контроллеру представление данных, запрашиваемых пользователем; вид, в свою очередь, обеспечивает представление данных, полученных из модели.

Storyboard – файл разметки, который предоставляет разработчику удобный набор инструментов для построения графического интерфейса. Несмотря на то, что файл использует язык разметки XML, программисту не нужно его знать, так как принцип работы построен на перетаскивании элементов на рабочий стол файла и указании свойств данного объекта через меню.

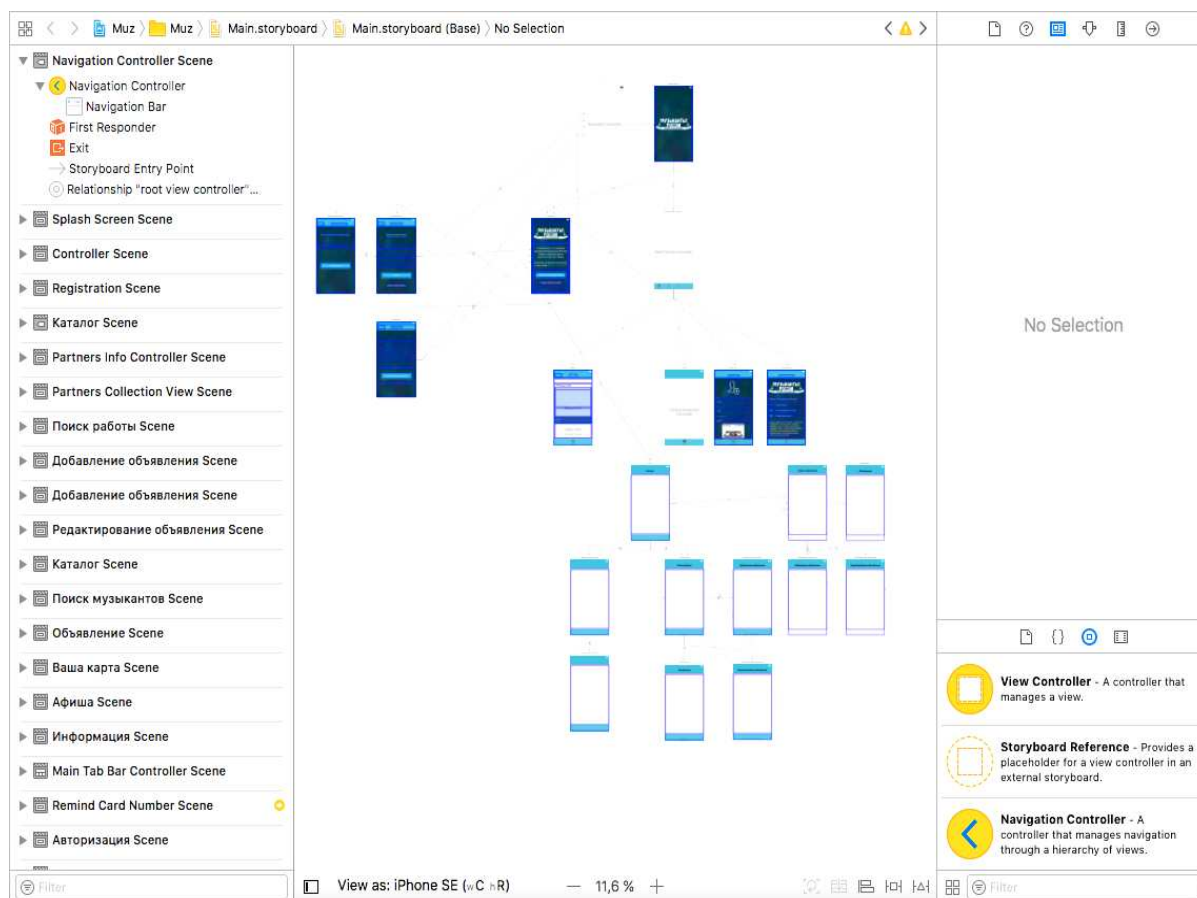


Рисунок 1. – Storyboard файл разметки

Данная технология имеет свои плюсы:

- наглядность и простота верстки;
- простое формирование переходов между экранами приложения;
- не нужно знать тип перехода, достаточно указать его идентификатор;
- SizeClasses – возможность задавать расположение элементов для различных экранов устройств

Однако есть и минусы:

- сложность при работе в команде, данные файлы плохо поддаются слиянию;
- большую часть кода анимации не вставить в данные файлы;
- большая нагрузка в случае увеличения размеров проекта;
- ограниченный выбор элементов разметки.

Как видно из рисунка 1, все переходы представлены стрелками между экранами, у каждого перехода есть свой идентификатор для доступа к ним. Также в правом нижнем углу расположено меню с выбором доступных элементов, однако список их ограничен.

Ручная разметка прописывается в файлах Controller-ов и View. Однако сделать это можно двумя способами:

- через фреймы;
- через систему Autolayout.

Работа с фреймами заключается в непосредственном указании координат расположения у элементов интерфейса, а также их размеров. Данный способ не является лучшим решением, так как он не предоставляет достаточной гибкости, т.е. при изменении ориентации экрана интерфейс не перестроится сам, придется предусмотреть и этот случай. Однако данный способ работает с ресурсоемкими элементами быстрее, чем Autolayout.

Autolayout – система работы с GUI, которая динамически вычисляет позицию и размер элементов, на основе определенных правил (constraints), заданных для этих элементов. Преимущество данной системы в гибкости, так как нет необходимости в подгонке размеров и положений элементов для различных экранов и состояний. Для облегчения работы с Autolayout существует множество открытых фреймворков: SnapKit, PureLayout и др.

Преимущество ручной разметки перед разметкой через Storyboard состоит в следующем:

- гибкость;
- быстрая работа;
- можно работать с расположением элементов в процессе работы с приложением;
- проще работать с анимацией;
- отсутствие ограничений в выборе элементов, так как можно написать свои элементы и использовать готовые.

Однако использование ручной разметки не дает наглядности, в приложении становится больше кода. Пример ручной разметки представлен в листинге 1.

Листинг 1 – пример ручной разметки

```
class VKCheckbox: UIView
{
    var line = VKCheckboxLine.Normal
    var button = UIButton()
    var checkmark = VKCheckmarkView()
    override init(frame: CGRect){
        super.init(frame: frame)
        self.setupView()
    }
    required init?(coder aDecoder: NSCoder){
        super.init(coder: aDecoder)
        self.setupView()
    }
    func setupView(){
        self.backgroundColor= UIColor.clear
        self.cornerRadius= 8
        self.borderWidth= 3
        self.borderColor= UIColor.darkGray
```

```
self.color = UIColor(red: 46/255, green: 119/255, blue: 217/255, alpha: 1)
self.checkmark.frame = self.bounds
self.checkmark.autoresizingMask = [.flexibleWidth, .flexibleHeight];
self.addSubview(self.checkmark)
self.button.frame = self.bounds
self.button.autoresizingMask = [.flexibleWidth, .flexibleHeight];
self.button.addTarget(self, action: #selector(VKCheckbox.buttonDidSelected), for: .touchUpInside)
self.addSubview(self.button)
}
override func layoutSubviews() {
    super.layoutSubviews()
    self.button.bounds = self.bounds
    self.checkmark.bounds = self.bounds
}
}
```

Заключение

Рассмотрены основные способы построения графического интерфейса для мобильных приложений под операционную систему iOS, а также их плюсы и минусы. Представлены примеры работы с каждой из систем.

ЛИТЕРАТУРА

1. Ruseller [Электронный ресурс] Частная коллекция качественных материалов для тех, кто делает сайты. – Режим доступа: <https://ruseller.com/lessons.php?id=666>. – Дата доступа: 26.09.2017.
2. Habrahabr [Электронный ресурс] Ручная верстка vsStoryboard/Nib. – Режим доступа: <https://habrahabr.ru/post/osbd/271355/>. – Дата доступа: 26.09.2017.
3. Apple Inc [Электронный ресурс] Understanding Auto Layout. – Режим доступа: <https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/AutoLayoutPG/index.html>. Дата доступа: 26.09.2017.
4. EnvatoTutsPlus [Электронный ресурс] iOS Fundamentals: Frames, Bounds and CGGeometry. – Режим доступа: <https://code.tutsplus.com/tutorials/ios-fundamentals-frames-bounds-and-cggeometry--cms-21196>. – Дата доступа: 26.09.2017.

УДК 004.021

**ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ
ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ «МУЗЫКАНТЫ РОССИИ»
ПОД ОПЕРАЦИОННУЮ СИСТЕМУ iOS**

А.В. ВЕЧЕРОВ*(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются проектирование базы данных для мобильного приложения «Музыканты России» под операционную систему iOS. Краткий анализ предметной области выявил необходимый набор сущностей. Делается вывод, что для разработки представленной базы данных удобно использовать СУБД MySQL, так как она является открытой и бесплатной, а также поддерживает множество типов данных необходимых при разработке.

Трудно представить хорошее клиент-серверное приложение без разработанной базы данных. Они позволяют удобно хранить большие объемы информации не на устройстве пользователя, а на удаленном сервере, который может эти данные предоставить по запросу от приложения.

Данная статья направлена на описание базы данных для разработанного приложения и методов ее проектирования.

Цель моделирования данных состоит в обеспечении разработчика концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Наиболее распространенным средством моделирования данных являются диаграммы «сущность-связь» (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.

Нотация ERD была впервые введена П. Ченом (Chen) и получила дальнейшее развитие в работах Баркера. Методология IDEF1, разработанная Т. Рэмей, основана на подходе П. Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме.

В настоящее время на основе совершенствования методологии IDEF1 создана ее новая версия – методология IDEF1X [1].

В результате краткого анализа предметной области выявлен необходимый набор сущностей:

- Partners (Партнеры);
- Partner_Cards (Карты пользователей);
- Search_Ad (Объявления о поиске музыкантов или группы);
- Work_Ad (Объявления о поиске работы);
- Click_Statistics (Статистика переходов пользователя по страницам информации о партнерах);
- Partner_Cities (Список городов);
- Partner_Categories (Список категорий);
- Partner_Likes (Рейтинг партнеров).

Связь – это ассоциирование двух или более сущностей. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

В любой связи выделяются два конца (в соответствии с существующей парой связываемых сущностей), на каждом из которых указывается:

- имя конца связи;
- степень конца связи (сколько экземпляров данной сущности связывается);
- обязательность связи (т.е. любой ли экземпляр данной сущности должен участвовать в данной связи) [2].

Для реализации приложения необходимо установить все связи между сущностями: необходимо рассмотреть всю информационную систему в совокупности и определить взаимное влияние сущностей.

Реляционная модель представляет базу данных в виде множества взаимосвязанных отношений (таблиц), которые используются для хранения информации об объектах, представленных в базе данных.

Концептуальная схема базы данных представлена на рисунке 1.

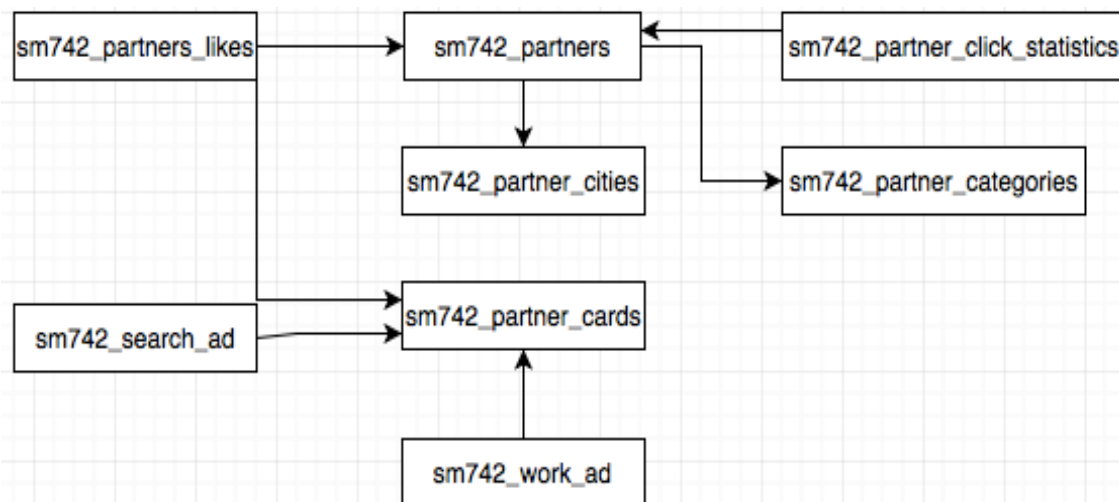


Рисунок 1. – Концептуальная схема базы данных

В соответствии выделенными выше сущностями определен набор необходимых таблиц базы данных, представленный в таблице 1.

Таблица 1. – Перечень таблиц базы данных

№ п/п	Название таблицы	Описание
1	Partners	Список всех партнеров
2	Partner_Cities	Список всех городов
3	Partner_Categories	Список всех категорий партнеров
4	Partner_Click_Statistics	Статистика переходов пользователя по страницам партнеров
5	Partners_Likes	Рейтинг партнеров
6	Partner_Cards	Список пользователей
7	Search_Ad	Список объявлений о поиске музыкантов или группы
8	Work_Ad	Список объявлений о работе

Важное достоинство баз данных – возможность хранить сгруппированные данные в разных таблицах и устанавливать связи между этими таблицами посредством ключей с последующим объединением этих данных в единую базу. Это позволяет снизить избыточность данных, которые хранятся, упростить организацию запросов для их получения.

Первичный ключ служит ограничителем целостности в рамках одной таблицы для идентификации, так как поле первичного ключа не может повторяться или быть пустым. Важное условие – каждая таблица может содержать только один первичный ключ.

Для организации связей между двумя и более таблицами используются вторичные ключи, которые служат ограничителями целостности связей нескольких таблиц, так как подчиненная таблица не может ссылаться на несуществующие записи главной таблицы (что позволяет строить целостные модели данных).

Существует три вида связей между таблицами:

- «один-ко-многим»;
- «один-к-одному»;
- «многие-ко-многим».

Связь «один-ко-многим» – самая распространенная для реляционных баз данных.

Для обеспечения ссылочной целостности в дочерней таблице создается внешний ключ, через который организовывается связь с дочерней таблицей.

Обязательное условие состоит в том, что внешний ключ по составу полей должен совпадать с первичным ключом родительской таблицы.

Первичные и внешние ключи отношений проектируемой базы данных представлены в таблице 2.

Таблица 2. – Первичные и внешние ключи отношений

№ п/п	Название таблицы	Первичный ключ	Внешние ключи
1	Partners	id	category city
2	Partner_Cities	id	
3	Partner_Categories	id	
4	Partner_Click_Statistics	id	partner_id
5	Partner_Cards	id	
6	Partners_Likes	id	partner_id user_id
7	Search_Ad	id	user_id
8	Work_Ad	id	user_id

Также важно выбрать систему управления базой данных.

Для разработки представленной базы данных удобно использовать СУБД MySQL, так как она является открытой и бесплатной, а также поддерживает множество типов данных, необходимых при разработке.

Заключение

Рассмотрены основные цели и принципы моделирования базы данных для приложения «Музыканты России».

Проанализированы основные сущности спроектированной базы данных.

ЛИТЕРАТУРА

1. Help4Students [Электронный ресурс] Методология моделирования данных ИС. – Режим доступа: [https://hlp4asp.wordpress.com/2011/10/13/методология-моделирования-данных-ис.](https://hlp4asp.wordpress.com/2011/10/13/методология-моделирования-данных-ис/) – Дата доступа: 27.09.2017.
2. CitForum [Электронный ресурс] Семантическое моделирование данных, ER-диаграммы. – Режим доступа: http://citforum.ru/database/osbd/glava_24.shtml. – Дата доступа: 27.09.2017.
3. Dev DocumentationMySQL[Электронный ресурс] MySQLWorkbench. – Режим доступа: <https://dev.mysql.com/doc/workbench/en/>. – Дата доступа: 27.09.2017.

УДК 004.021

**РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ ИНТЕРНЕТ-МАГАЗИНОМ,
ВЗАИМОДЕЙСТВИЕ СИСТЕМЫ С ПОЛЬЗОВАТЕЛЕМ****А.В. ГУБКО****(Представлено: Д.В. ПЯТКИН)**

Представлен практический способ создания простой и интуитивно понятной системы управления для интернет-магазина. Цель работы состояла в написании приложения для упрощения взаимодействия пользователя с базой данных при работе с наполнением интернет-магазина, для упрощения пользования самим интернет-магазином, а также в создании простейшего и понятного интерфейса управления и разработка защиты от несанкционированного доступа со стороны. Задача решалась посредством скриптового языка сценариев PHP.

В настоящее время человек старается оцифровать как можно больший объем информации, что позволит иметь к ней доступ с любого цифрового устройства в любой точке мира, имея лишь доступ к сетевым ресурсам. Это позволяет реализовать самые обычные и привычные всем действия более быстро. Например, совершать покупки. Но мало кто задумывался, что за системой, занимающейся продажами стоит сложная логика обработки заказов и демонстрации товаров. Именно для управления заказами и наполнением магазина занимается система управления. Это значит, что данная система оперирует большими объемами данных, включая платежные данные, личные данные покупателей, а также зачастую данные самого магазина и данные о товарах. Из этого следует, что рассматриваемая система должна отвечать некоторым требованиям к безопасности и предоставлять доступ узкому кругу лиц.

С учетом растущих требований у населения возрастают также требования к интерфейсу разрабатываемых продуктов. Как и в любом веб-приложении, интерфейс позволяет пользователю получать требуемую информацию в приемлемом виде.

Выбор технологий

Данная часть приложения является неотъемлемой частью интернет-магазина, следовательно, она также является веб-приложением. Именно поэтому разработка велась на языке PHP. Данный язык позволил реализовать работу с доступом к информации в базе данных. Данный язык предоставляет возможность быстро задать подключение к базе данных.

Помимо языка PHP, для улучшения внешнего вида интерфейса использовались такие языки, как Javascript и CSS.

CSS – формальный язык описания внешнего вида документа, написанного с помощью языка разметки. Служит в основном для создания каскадных таблиц стилей.

Javascript – язык программирования, придающий страницам интерактивность; служит для создания приложений и частей приложений.

Проектирование интерфейса

При разработке интерфейса, для удобства использования был применен способ группировки задач по блокам с определенной тематикой. Данный вид группировки позволяет упростить и ускорить поиск блоков, а также систематизировать данные на экране, для лучшей читабельности и понимания.

Сравнение доступа через систему и через базу

Для добавления информации в интернет магазин напрямую через базу данных пользователю в определенных случаях необходимо добавлять связи и описание сразу в большое количество таблиц, как показано на рисунке 1.

Как видно из рисунка, для добавления информации необходимо задействовать большое количество таблиц. Также при оформлении заказа, когда информация требует обновления, придется искать необходимую информацию во всех таблицах. Это существенно осложняет работу с наполнением интернет-магазина и с самим процессом работы с ним. Именно для этих целей и разрабатывается система управления содержимым интернет-магазина (рисунок 2).

На рисунке 2 можно видеть, что тот же процесс добавления данных, но используя систему управления, становится гораздо проще и быстрее. Простота достигается тем, что все связи задаются заранее через программный код, и пользователю не надо отслеживать все таблицы на предмет наличия данных, система сделает все сама. Пользователю остается только ввести необходимые данные для добавления или выбрать данные, требующие изменения. Доступ напрямую через базу очень сложен, и даже если пользователь обладает всеми необходимыми данными о структуре базе данных, данный процесс все равно остается достаточно длительным. Также никто не застрахован от человеческих ошибок и обычной

невнимательности, что может повлечь за собой потерю данных, и в итоге – возможную потерю прибыли от интернет-магазина.

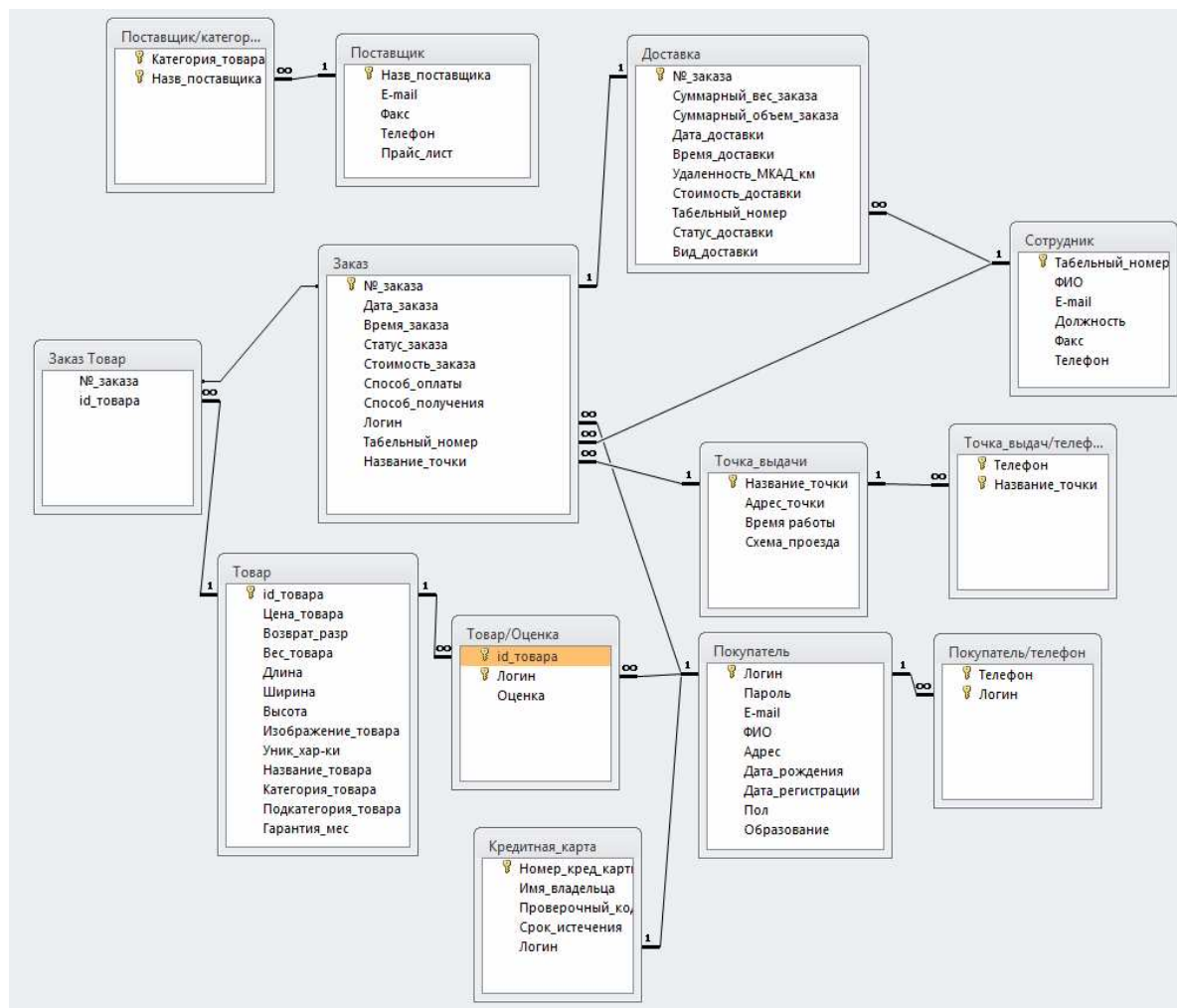


Рисунок 1. – Количество таблиц в базе данных интернет магазина

Новая сделка

☰

Связать с интернет-магазином: ☒

☰

О сделке

☰

Название:

✕ | ✎

Стадия сделки:

▼

Валюта:

▼

Рисунок 2. – Добавление информации в базу через интерфейс

Защита данных от несанкционированного доступа

Так как данная система оперирует всеми данными, которые находятся в базе данных (данные пользователей, счета, номера карт, пароли), то при их утрате могут возникнуть довольно большие проблемы.

Исходя из вышесказанного, можно сделать вывод, что данные подлежат защите. Самым простым методом защиты может стать закрытие прямого доступа к системе, чтобы никто не смог получить доступ к данным. Но тогда возникает ситуация, что доступ должен быть у определенного круга лиц, которые могут работать в системе. Решением данной проблемы является создание группы пользователей, которой выдан доступ для работы с данной системой. Это позволяет ограничить доступ для сторонних лиц и предоставить доступ к системе выделенному кругу лиц. Но иногда этого может быть недостаточно, и сторонние лица могут завладеть правами доступа к системе. В этом случае потребуется вторая ступень защиты, которая заключается в шифровании паролей и номеров карт, чтобы никто не имел возможности видеть «открытые» данные о паролях и номерах карт. Также сама система может являться ступенью защиты, так как с базой данных взаимодействует не сам пользователь, а сама система. Защита данных с помощью системы объясняется тем, что при взаимодействии с базой пользователь может ввести некорректные данные и тем самым повлечет как минимум наличие некорректных данных, а как максимум – очистку или даже удаление базы данных. Система же должна проводить проверку вводимых данных на корректность, запрашивать повторный ввод данных либо производить редактирование самостоятельно.

Взаимодействие с пользователем

Система должна взаимодействовать с пользователем для облегчения поиска информации и увеличения производительности. В случае если система позволяет оформлять заказы, пользователь должен видеть, что поступил заказ. Система должна уведомлять об этом. В данном случае пользователю не придется самому просматривать данные в поисках заказа, так как система должна автоматически сделать это сама. Это пример сложного взаимодействия с системой, однако есть и более простые примеры. В системе имеются блоки, описывающие данные из таблиц базы данных, и пользователь интуитивно понимает, с какой частью системы он работает в данный момент. Также во взаимодействии с пользователем существенную роль играет наличие интерфейса, который позволяет обеспечить удобный и понятный процесс работы в самой системе.

Заключение

Разработка системы управления является неотъемлемой частью создания интернет-магазина, так как позволяет не только упростить и ускорить работу в самой системе, но и защитить данные пользователя, клиентов и данные самого магазина. Разработка же интерфейса в существенной мере упрощает понимание элементов системы, что также облегчает дальнейшую работу.

ЛИТЕРАТУРА

1. PHP. Hypertext Preprocessor [Электронный ресурс]. – Режим доступа: <http://php.net/manual/ru/intro-what-is.php>. – Дата доступа: 28.09.2017.
2. Материал из Википедии – свободной энциклопедии CSS [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/CSS>. – Дата доступа: 28.09.2017.
3. Javascriptто это такое? | Все что вы хотели знать о хостинге и сайтостроении [Электронный ресурс]. – Режим доступа: <http://articles-hosting.ru/329/javascript-%E2%80%93-chto-eto-takoe.html>. – Дата доступа: 28.09.2017.

УДК 004.021

**ЭКОНОМИЧЕСКАЯ СОСТАВЛЯЮЩАЯ РАЗРАБОТКИ ИНТЕРНЕТ-МАГАЗИНА,
ОЦЕНКА ВЛИЯНИЯ ИНФОРМАЦИОННОЙ СРЕДЫ НА ЭКОНОМИКУ****А.В. ГУБКО****(Представлено: Д.В. ПЯТКИН)**

Представлена информация об экономической части разработки интернет магазина. Также представлена информация об основных стадиях разработки приложения и их экономической составляющей. Проводится сравнительный анализ интернет магазинов с обычными магазинами. Сравниваются их достоинства и недостатки. Оценивается экономическая эффективность.

В настоящее время стремительными темпами развивается сфера интернет-торговли. Все большее количество людей предпочитают совершать покупки онлайн, не выходя из дома, даже если учесть тот факт, что покупатель не может полностью оценить товар перед покупкой, существует риск быть обманутым. Однако все это не отпугивает людей от совершения покупок через интернет. При покупке товаров через интернет покупатели могут руководствоваться различными сервисами, предоставляющими возможность обмена, впечатлениями различных реальных пользователей по поводу того или иного интернет-магазина.

Стадии разработки:

С экономической точки зрения разработка любого приложения, будь то магазин или игра, делится на 4 важнейших этапа:

- 1) проектирование;
- 2) реализация;
- 3) тестирование;
- 4) доработка.

На этапе *проектирования* проводится анализ для дальнейшей разработки. Уже на этом этапе закладываются основные требования и определяются примерные объемы работ и затраты на них.

К этапу *реализации* относится непосредственно сама разработка продукта. Данный этап является самым долгим и объемным, также он включает в себя наибольшие затраты на саму разработку.

На этапе *тестирования* проводится тестирование разработанного продукта с целью выявления неточностей и погрешностей в логике или методах решения поставленных задач. Этап доработки включает финальные испытания и изменения системы, а также работы по приемке-сдаче продукта в оборот. Затраты на каждом из этапов проектирования можно увидеть на рисунке 1.

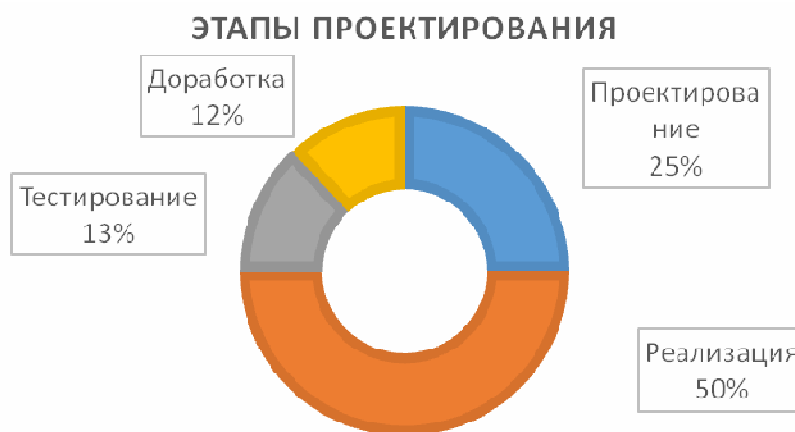


Рисунок 1. – Затраты на этапах проектирования

Расчет стоимости продукта

В стоимость любого продукта включена стоимость его создания. В данном случае, программным продуктом является приложение, интернет-магазин. Расчет стоимости должен учитывать следующие статьи затрат:

- 1) *материалы и комплектующие* – расходные материалы, потраченные при создании продукта;

- 2) *электроэнергия* – так как продуктом является приложение, то при его разработке используют компьютеры, потребляющие электричество;
 - 3) *основная заработная плата исполнителей* – затраты на выдачу заработной платы сотрудникам, работающим над проектом;
 - 4) *дополнительная заработная плата* – затраты на оплату премий, рисков, отпусков и прочее;
 - 5) *отчисления на социальные нужды* – затраты на налоговые сборы, в фонд защиты населения, страхование, прочее;
 - 6) *амортизация* – затраты на непредвиденные обстоятельства;
 - 7) *расходы на спецоборудование* – затраты, связанные с приобретением специализированного оборудования;
 - 8) *накладные расходы* – расходы на общехозяйственные нужды;
 - 9) *прочие прямые расходы* – затраты на подготовку и приобретение различного рода информации.
- Основные затраты на разрабатываемый продукт, вне зависимости от сроков разработки, приходятся на выдачу заработной платы сотрудникам.

График распределения затрат можно наблюдать на рисунке 2.

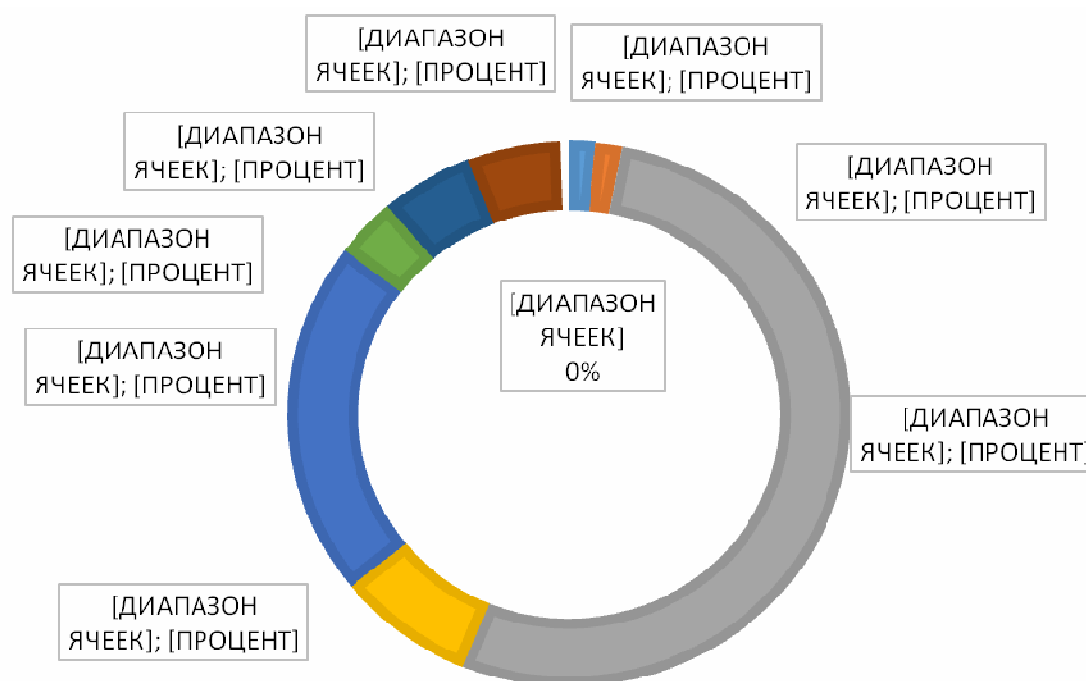


Рисунок 2. – структура затрат на разработку

В каждом отдельном случае данные могут меняться, однако при любых обстоятельствах основными затратами будет являться заработная плата сотрудникам, выполняющим работы по разработке продукта. Эти факторы следует учитывать при планировании каждого отдельного проекта, и при планировании времени, за которое планируется погасить расходы и выйти в прибыль.

Сравнительный анализ интернет магазинов с обычными магазинами

В данный момент появилась тенденция, согласно которой интернет-магазины становятся более популярными, чем обычные магазины. Это существенно увеличивает объемы продаж, а соответственно и прибыль интернет-магазинов.

На рисунке 3 можно видеть, насколько увеличились объемы продаж в интернет-магазинах за последние годы. В это же время популярность обычных магазинов начинает падать, и чтобы не потерять клиентов, обычные магазины начинают внедрять интернет-ресурсы в свою деятельность. Например, онлайн-заказ продуктов и онлайн-доставка. Однако не стоит забывать, что существенным преимуществом обычных магазинов является наглядное представление товара и возможность самостоятельного выбора.

Достоинства и недостатки:

Каждый из видов магазинов имеет свои плюсы и минусы.

Достоинства интернет-магазина:

+ малый стартовый капитал;

- + сравнительно недорогое содержание магазина;
- + проведение аналитики;
- + спрос, обусловленный растущим числом потребителей;
- + более низкая цена, за счет малых затрат на содержание;

Достоинства обычного магазина:

- + возможность личной оценки товара;
- + моментальность покупки;
- + возможность возврата;
- + большее доверие клиентов.

В данном случае достоинства одного вида магазина являются недостатком магазина другого вида.



Рисунок 3. – Объемы продаж интернет магазинов

Оценка экономической эффективности:

Как было упомянуто ранее, интернет-магазины приобретают все большую популярность, в то время как обычные магазины ее теряют. Этот факт существенно влияет на объемы продаж и объемы прибыли. Именно поэтому разработка и содержание интернет-магазина является более экономически выгодным, нежели открытие обычного магазина.

Заключение

Экономика развивается совместно с информационными технологиями. Разработка интернет-магазинов является экономически выгодным, т.к. в данный момент все больше клиентов хотят упростить жизнь, и сделать ее комфортной. С помощью интернет-магазинов можно экономит не только деньги, но и время. Интернет-магазины позволяют избавиться от утомительного ожидания в очередях. Однако помимо всего, у интернет-магазинов имеется и ряд недостатков, основным из которых является отсутствие возможности полного просмотра приобретаемого товара.

ЛИТЕРАТУРА

1. Insales.ru [Электронныйресурс]. – Режим доступа: <https://www.insales.ru>. – Дата доступа: 28.09.2017.
2. AvenueSoft.ru [Электронныйресурс]. – Режим доступа: http://avenuesoft.ru/shop/internet_magazin_vs_offline_magazin.html. – Дата доступа: 28.09.2017.
3. Richpro.ru – финансовый журнал [Электронныйресурс]. – Режим доступа:<http://richpro.ru>. – Дата доступа: 28.09.2017.

УДК 004.932.72

**ОБНАРУЖЕНИЕ ЛИЦ НА ИЗОБРАЖЕНИЯХ
МЕТОДОМ ГИСТОГРАММЫ НАПРАВЛЕННЫХ ГРАДИЕНТОВ****И.И. ИВАНОВ***(Представлено: Е.С. ГАТИХО)*

Рассматривается возможность обнаружения и локализации лиц на изображениях с помощью применения метода, основанного на построении гистограммы направленных градиентов в совокупности с методом скользящего окна.

В задачах, связанных с машинным зрением, одной из самых ресурсоемких подзадач остается обнаружение и локализация объекта на изображении для последующей обработки выделенного участка алгоритмами классификации. Метод, основанный на построении гистограммы направленных градиентов, является одним из самых эффективных решений, используемых для обнаружения объектов.

Основной идеей данного алгоритма является допущение, что внешний вид и форма объекта на фрагменте изображения могут быть описаны распределением градиентов интенсивности. При описании фрагмента изображения оно разбивается на несколько небольших участков, называемых ячейками. В ячейках вычисляются гистограммы направленных градиентов внутренних пикселей. Таким образом, конечный дескриптор, являющийся комбинацией всех гистограмм, отображает направления перехода от света к темноте по всему изображению, показывая упрощенное представление объектов на нем, независимо от его яркости [1]. Пример гистограммы направленных градиентов представлен на рисунке 1.



Рисунок 1. – Пример гистограммы направленных градиентов

Как видно, объекты на гистограмме упрощенно представляются множеством направлений градиентов, что позволяет выделить их формы и, соответственно, примерный внешний вид искомого объекта. Этого вполне достаточно для его легкой локализации с использованием канонического шаблона, составленного с использованием некоторой выборки изображений, непосредственно содержащих искомым объект в необходимой проекции.

При вычислении градиентов производится свертка изображения с ядрами $[-1, 0, 1]$ и $[-1, 0, 1]^T$, в результате чего образуются две матрицы D_x и D_y производных вдоль осей x и y соответственно. Эти матрицы используются для вычисления углов и модулей градиентов в каждой точке изображения.

Каждому интервалу ставится в соответствие бин гистограммы. Тогда гистограмма ячейки заполняется так, что величина градиента в каждой внутренней его точке добавляется к величине бина, соответствующего интервалу, содержащему угол данного градиента [2].

Дескриптор гистограммы ориентированных градиентов имеет несколько преимуществ над другими дескрипторами. Поскольку он работает локально, метод поддерживает инвариантность геометриче-

ских и фотометрических преобразований, за исключением ориентации объекта. Подобные изменения появятся только в больших фрагментах изображения.

Для непосредственного обнаружения и локализации объекта разумно воспользоваться методом скользящего окна, заключающегося в отыскании такого участка изображения, который наиболее похож на известный шаблон объекта. Пример шаблона человеческого лица, составленного с помощью набора изображений лиц в фронтальной проекции, представлен на рисунке 2.



Рисунок 2. – Пример шаблона человеческого лица

Метод скользящего окна использует проход изображения областью, последовательно меняющей свое положение и размер, а также нахождение некоторого коэффициента схожести участка изображения, над которым на данном шаге находится текущее «окно» и шаблона искомого объекта.

Также иногда применяются некоторые модификации и улучшения метода скользящего окна в виде, например, динамического шага перемещения или коэффициента масштабирования окна в зависимости от содержимого изображения в соседних областях, что позволяет исключить излишние вычислительные операции и ускорить процесс обнаружения [2].

Также операции, связанные с процессом обнаружения, могут быть разбиты на несколько независимых друг от друга частей и произведены в несколько вычислительных потоков, что только ускорит работу алгоритма.

Следует отметить, что обнаружение таких объектов, как человеческие лица, не требует наличия множества деталей на изображении, что, безусловно, дает возможность масштабировать крупные изображения, многократно выигрывая в производительности, не теряя процента обнаружения, если не требуется локализация лиц относительно малых размеров.

Исходя из вышеописанного можно сделать следующие **выводы**. Применение метода обнаружения объектов на изображении, основанного на построении гистограммы направленных градиентов для обнаружения и локализации лиц, позволяет решать поставленную задачу с высокой точностью, не прибегая к высоким затратам вычислительных ресурсов. Это дает возможность построения систем обнаружения лиц не только на статичных изображениях, но и на множестве кадров, образующих видеопоток. Более того, с применением вышеописанных оптимизаций, модификаций и ухищрений, нацеленных на ускорение работы метода скользящего окна, метод, использующий построение гистограммы направленных градиентов, позволяет создавать легкие и эффективные системы обнаружения лиц даже для мобильных платформ, ограниченных в вычислительных мощностях, без ощутимого снижения эффективности.

Вкупе с невысокой чувствительностью к силе освещения, метод, основанный на построении гистограммы направленных градиентов, является одним из самых эффективных и популярных в сфере обнаружения и локализации объектов, задаваемых единым каноничным шаблоном, сгенерированным на небольшом множестве примеров изображений.

ЛИТЕРАТУРА

1. Гистограмма направленных градиентов [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Гистограмма_направленных_градиентов. – Дата обращения: 01.09.2017.
2. Южаков, Г.Б. Алгоритм быстрого построения дескрипторов изображения, основанных на технике гистограмм ориентированных градиентов / Г.Б. Южаков // ТРУДЫ МФТИ. – 2013. – Т. 5, № 3. – С. 87.

УДК 004.021

КЛАССИФИКАЦИЯ ДАННЫХ МЕТОДОМ ОПОРНЫХ ВЕКТОРОВ

И.И. ИВАНОВ

(Представлено: Е.С. ГАТИХО)

Анализируется классификация данных методом опорных векторов: описываются причины использования, описание алгоритма, его плюсы и минусы. Применение метода опорных векторов позволяет создать высокоточные системы классификации данных, имеющих место в статистическом анализе и методах машинного зрения.

Классификация данных – одна из центральных задач машинного обучения. В данном направлении интенсивно применяются методы оптимизации и аналитической геометрии. Задача классификации данных состоит в нахождении ответа на вопрос, к какому из определенных классов будут принадлежать новые наблюдаемые данные, основываясь на уже имеющемся наборе данных, классы которых известны. Каждый объект данных представляется как вектор (точка) в n -мерном пространстве (упорядоченный набор n чисел). Каждая из этих точек принадлежит только одному из определенных классов. Вопрос состоит в том, можно ли разделить точки гиперплоскостью размерности $n - 1$. Это типичный случай линейной делимости. Искомых гиперплоскостей может быть множество, поэтому полагают, что максимизация зазора между классами способствует более уверенной классификации. То есть, можно ли найти такую гиперплоскость, чтобы расстояние от нее до ближайшей точки было максимальным. Это эквивалентно тому, что расстояние между двумя ближайшими точками, лежащими по разные стороны гиперплоскости, максимально. Если такая гиперплоскость существует, она называется оптимальной разделяющей гиперплоскостью, а соответствующий ей линейный классификатор называется оптимально разделяющим классификатором [1].

Пример иллюстрации оптимальной и неоптимальных разделяющих плоскостей в рамках абстрактного набора данных, заданного множеством векторов размерности 2, приведен на рисунке 1.

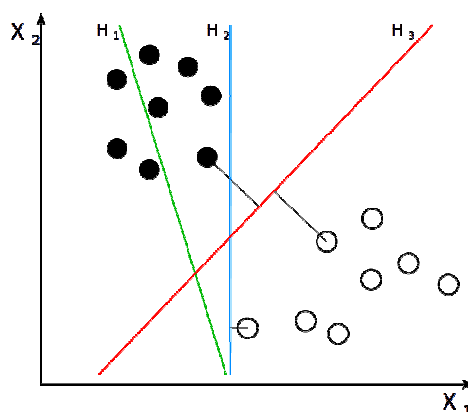


Рисунок 1. – Пример оптимальных и неоптимальных разделяющих плоскостей

При описании задачи полагается, что точки имеют вид:

$$\{(c_1 x_1), (c_2 x_2), \dots, (c_p x_p)\},$$

где c_i принимает значение 1 или -1 , в зависимости от того, какому классу принадлежит точка x_i .

Каждое x_i – это n -мерный вещественный вектор, обычно нормализованный значениями $[0,1]$ или $[-1,1]$. Если точки не будут нормализованы, то точка с большими отклонениями от средних значений координат точек слишком сильно повлияет на классификатор. Можно рассматривать это как учебную коллекцию, в которой для каждого элемента уже задан класс, к которому он принадлежит.

Для того чтобы метод опорных векторов классифицировал их таким же образом, строится разделяющая гиперплоскость, имеющая вид:

$$w \cdot x - b = 0.$$

Вектор w – перпендикулярен к разделяющей гиперплоскости.

Параметр $\frac{b}{\|w\|}$ равен по модулю расстоянию от гиперплоскости до начала координат.

Если параметр b равен нулю, гиперплоскость проходит через начало координат, что ограничивает решение. Так как производится поиск оптимального разделения, выделяются опорные векторы и гипер-

плоскости, параллельные оптимальной и ближайшие к опорным векторам двух классов. Можно показать, что эти параллельные гиперплоскости могут быть описаны следующими уравнениям (с точностью до нормировки):

$$w \cdot x - b = 1,$$

$$w \cdot x - b = -1.$$

Если обучающая выборка линейно разделима, то мы можем выбрать гиперплоскости таким образом, чтобы между ними не лежала ни одна точка обучающей выборки, и затем максимизировать расстояние между гиперплоскостями [2].

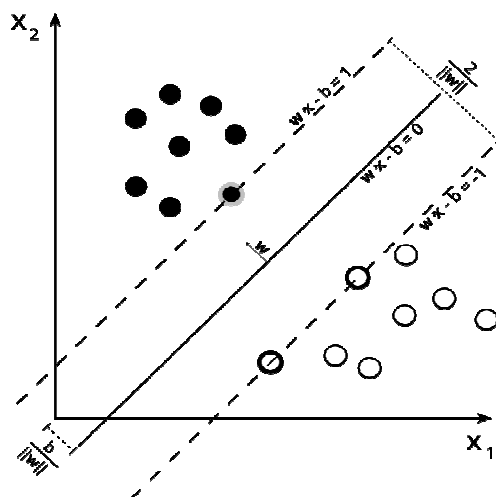


Рисунок 2. – Пример оптимальной гиперплоскости с выделенными опорными векторами

Ширину полосы между ними легко найти из соображений геометрии, она равна $\frac{2}{\|w\|}$. Таким образом, наша задача минимизировать $\|w\|$. Чтобы исключить все точки из полосы, мы должны убедиться для всех i , что и

$$w \cdot x_i - b \geq 1, \quad c_i = 1,$$

$$w \cdot x_i - b \leq -1, \quad c_i = 1.$$

Это может быть также записано в виде:

$$c_i (w \cdot x_i - b) \geq 1, 1 \leq i \leq n.$$

Метод опорных векторов обладает следующими преимуществами [2]:

- наиболее быстрый метод нахождения решающих функций;
- метод сводится к решению задачи квадратичного программирования в выпуклой области, которая всегда имеет единственное решение;
- метод находит разделяющую полосу максимальной ширины, что позволяет в дальнейшем осуществлять более уверенную классификацию;
- возможность классификации среди множества классов;
- малые временные затраты непосредственно на классификацию данных.

Однако, как и любой алгоритм, метод опорных векторов не лишен и своих недостатков, главным из которых является чувствительность к шумам и стандартизации данных.

Исходя из вышесказанного, можно сделать *вывод*, что применение метода опорных векторов позволяет создать высокоточные системы классификации данных, имеющих место в статистическом анализе и методах машинного зрения.

ЛИТЕРАТУРА

1. Support vector machine [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Support_vector_machine. – Дата доступа: 02.09.2017.
2. Машина опорных векторов [Электронный ресурс]. – Режим доступа: <http://www.machinelearning.ru/wiki/index.php?title=SVM>. Дата обращения: 02.09.2017.

УДК 004.056.55

СИММЕТРИЧНЫЕ И АСИММЕТРИЧНЫЕ ШИФРЫ

Р.Ю. КАРАБАНОВ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматриваются и сравниваются два самых распространенных типа шифров – симметричные и асимметричные. Выделены их достоинства и недостатки. Для криптографического закрытия и последующего расшифровывания передаваемой информации используются открытый и секретный ключи.

Шифр (от фр. chiffre «цифра» от араб. صفر, *sifr* «ноль») – какая-либо система преобразования текста с секретом (ключом) для обеспечения секретности передаваемой информации. Шифр может представлять собой совокупность условных знаков (условная азбука из цифр или букв) либо алгоритм преобразования обычных цифр и букв.

По количеству используемых ключей шифры разделяются на *симметричные* (используют один ключ для шифрования и дешифрования) и *асимметричные* (используют два различных ключа) [1].

Симметричные шифры

Исторически первыми появились симметричные криптографические системы. В симметричной криптосистеме шифрования используется один и тот же ключ для зашифровывания и расшифровывания информации. Это означает, что любой, кто имеет доступ к ключу шифрования, может расшифровать сообщение.

Соответственно, с целью предотвращения несанкционированного раскрытия зашифрованной информации все ключи шифрования в симметричных криптосистемах должны держаться в секрете. Именно поэтому симметричные криптосистемы называют *криптосистемами с секретным ключом* – ключ шифрования должен быть доступен только тем, кому предназначено сообщение. Симметричные криптосистемы называют еще *одноключевыми* криптографическими системами, или *криптосистемами с закрытым ключом*.

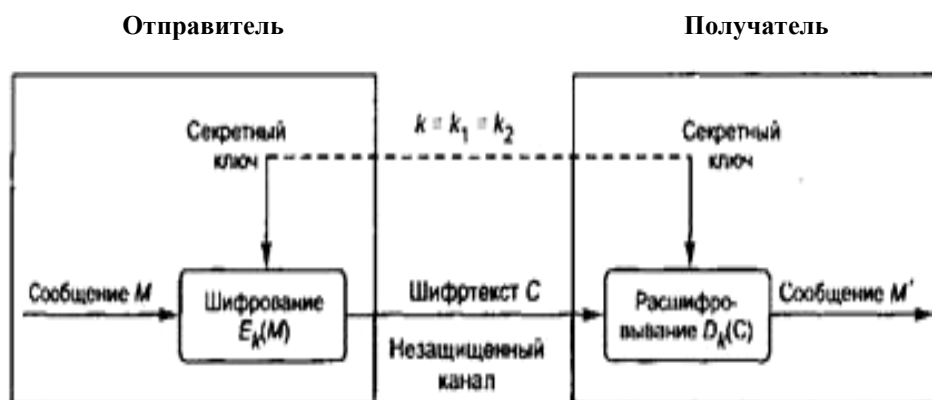


Схема симметричной криптосистемы шифрования

Данные криптосистемы характеризуются наиболее высокой скоростью шифрования, и с их помощью обеспечиваются как конфиденциальность и подлинность, так и целостность передаваемой информации. Конфиденциальность передачи информации с помощью симметричной криптосистемы зависит от надежности шифра и обеспечения конфиденциальности ключа шифрования [2].

С одной стороны, такая схема имеет те недостатки, что необходимо кроме открытого канала для передачи шифрограммы наличие также секретного канала для передачи ключа, а кроме того, при утечке информации о ключе, невозможно доказать, от кого из двух корреспондентов произошла утечка.

С другой стороны, среди шифров именно этой группы есть единственная в мире схема шифровки, обладающая абсолютной теоретической стойкостью. Все прочие можно расшифровать хотя бы в принципе. Такой схемой является обычная шифровка с ключом, длина которого равна длине сообщения. При этом ключ должен использоваться только раз. Любые попытки расшифровать такое сообщение беспо-

лезны, даже если имеется априорная информация о тексте сообщения. Осуществляя подбор ключа, можно получить в результате любое сообщение [3].

Асимметричные шифры

Асимметричные криптографические системы были разработаны в 1970-х годах. Принципиальное отличие асимметричной криптосистемы от криптосистемы симметричного шифрования состоит в том, что для шифрования информации и ее последующего расшифровывания используются различные ключи:

- *открытый ключ* K используется для шифрования информации, вычисляется из секретного ключа k ;
- *секретный ключ* k используется для расшифровывания информации, зашифрованной с помощью парного ему открытого ключа K .

Эти ключи различаются таким образом, что с помощью вычислений нельзя вывести секретный ключ k из открытого ключа K . Поэтому открытый ключ K может свободно передаваться по каналам связи.

Асимметричные системы называют также двухключевыми криптографическими системами, или криптосистемами с открытым ключом.



Схема асимметричной криптосистемы шифрования

Для криптографического закрытия и последующего расшифровывания передаваемой информации используются открытый и секретный ключи получателя B сообщения.

В качестве ключа зашифровывания должен использоваться открытый ключ получателя, а в качестве ключа расшифровывания – его секретный ключ.

Секретный и открытый ключи генерируются попарно. Секретный ключ должен оставаться у его владельца и быть надежно защищен от НСД (аналогично ключу шифрования в симметричных алгоритмах). Копия открытого ключа должна находиться у каждого абонента криптографической сети, с которым обменивается информацией владелец секретного ключа [5].

Сравнение шифров

Достоинства симметричной криптосистемы:

- скорость;
- простота реализации (за счёт более простых операций);
- меньшая требуемая длина ключа для сопоставимой стойкости;
- изученность (за счёт большего возраста);

Недостатки симметричной криптосистемы:

- сложность управления ключами в большой сети;
- сложность обмена ключами. Для применения необходимо решить проблему надёжной передачи ключей каждому абоненту, так как нужен секретный канал для передачи каждого ключа обеим сторонам [6].

Достоинства асимметричной криптосистемы:

- не нужно предварительно передавать секретный ключ по надёжному каналу;
- только одной стороне известен ключ дешифрования, который нужно держать в секрете (в симметричной криптографии такой ключ известен обеим сторонам и должен держаться в секрете обеими);
- в больших сетях число ключей в асимметричной криптосистеме значительно меньше, чем в симметричной.

Недостатки асимметричной криптосистемы:

- в алгоритм сложнее внести изменения;
- более длинные ключи – ниже приведена таблица, сопоставляющая длину ключа симметричного алгоритма с длиной ключа RSA с аналогичной криптостойкостью;
- шифрование-расшифровывание с использованием пары ключей проходит на два-три порядка медленнее, чем шифрование-расшифрование того же текста симметричным алгоритмом;
- требуются существенно большие вычислительные ресурсы, поэтому на практике асимметричные криптосистемы используются в сочетании с другими алгоритмами [5].

ЛИТЕРАТУРА

1. Шифр [Электронный источник]. – 2017. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80>. – Дата доступа: 24.09.2017.
2. Симметричные криптосистемы шифрования [Электронный источник]. – 2017. – Режим доступа: <http://yupn.ru/193/symmetric-encryption-system>. – Дата доступа: 24.09.2017.
3. Типы шифров [Электронный источник]. – 2017. – Режим доступа: http://studbooks.net/2243124/informatika/typy_shifrov. – Дата доступа: 24.09.2017.
4. Асимметричные криптосистемы шифрования [Электронный источник]. – 2017. – Режим доступа: <http://yupn.ru/197/asymmetric-encryption-system/>. – Дата доступа: 24.09.2017.
5. Криптосистема с открытым ключом [Электронный источник]. – 2017. – Режим доступа: https://ru.wikipedia.org/wiki/Криптосистема_с_открытым_ключом. – Дата доступа: 24.09.2017.
6. Симметричные криптосистемы [Электронный источник]. – 2017. – Режим доступа: https://ru.wikipedia.org/wiki/Симметричные_криптосистемы. – Дата доступа: 24.09.2017.

УДК 004.056.55

ПРОБЛЕМАТИКА ПОПУЛЯРНЫХ АЛГОРИТМОВ ШИФРОВАНИЯ

Р.Ю. КАРАБАНОВ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматриваются самые популярные алгоритмы шифрования и их проблемы в современном информационном мире. Приводятся примеры проблем алгоритмов, в частности опыт взлома и их анализ.

В современном мире приводятся множество требований к алгоритму шифрования, начиная от криптостойкости и заканчивая временем для шифрования/дешифрования данных. Однако самым главным критерием оценки проблемы будем считать факт взлома алгоритма до этого. В данной работе остановимся на трех самых популярных на данное время алгоритмах – AES, DES, RSA. Именно эти три алгоритма реализованы в стандартной библиотеке «javax.crypto» в языке программирования Java – самом популярном объектно-ориентированном языке программирования на 2016 год [1]. Также следует отметить, что данные алгоритмы были введены в общее пользование до нынешнего десятилетия [2–4], а следовательно, не могут, в эру быстро развивающихся информационных технологий называться современными.

1. **DES** (англ. data encryption standard) – алгоритм для симметричного шифрования, разработанный фирмой IBM и утвержденный правительством США в 1977 году как официальный стандарт (FIPS 46-3). Размер блока для DES равен 64 бита. В основе алгоритма лежит сеть Фейстеля с 16 циклами (раундами) и ключом, имеющим длину 56 бит. Алгоритм использует комбинацию нелинейных (S-блоки) и линейных (перестановки E, IP, IP-1) преобразований [4].

Основная проблема. Так как длина ключа равна 56 битам, существует 2^{56} возможных ключей. Сегодня такая длина ключа недостаточна, поскольку допускает успешное применение лобовых атак. Альтернативой DES можно считать тройной DES, IDEA, а также алгоритм Rijndael, принятый в качестве нового стандарта на алгоритмы симметричного шифрования.

Также без ответа пока остается вопрос, возможен ли криптоанализ с использованием существующих характеристик алгоритма DES. Основой алгоритма являются восемь таблиц подстановки, или S-boxes, которые применяются в каждой итерации. Имеет место опасность, что эти S-boxes конструировались таким образом, что криптоанализ возможен для взломщика, который знает слабые места S-boxes. В течение многих лет обсуждалось как стандартное, так и неожиданное поведение S-boxes, но все-таки никому не удалось обнаружить их фатально слабые места [5].

Вывод: данный алгоритм можно легко взломать с помощью перебора.

2. **Advanced Encryption Standard (AES)** – симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством США по результатам конкурса AES. Этот алгоритм хорошо проанализирован и сейчас широко используется, как это было с его предшественником DES. Национальный институт стандартов и технологий США (англ. National Institute of Standards and Technology, NIST) опубликовал спецификацию AES 26 ноября 2001 года после пятилетнего периода, в ходе которого были созданы и оценены 15 кандидатур. 26 мая 2002 года AES был объявлен стандартом шифрования [3]. Суть AES в том, что любая «лобовая атака» на защищенные данные, то есть подбор всех возможных паролей, в перспективе очень сильно растягивается. Если представить, что взломщик располагает огромными ресурсами, то есть целой коллекцией суперкомпьютеров, то при усердном старании доступ к зашифрованным данным он мог бы получить через десятки лет. Если же в его распоряжении ничего этого нет, то взлом AES займет астрономически долгое время [6].

Основная проблема. Криптографы нашли метод взлома ключей AES-шифра ещё в 2011 году во время конференции по криптографии Crypto 2011, которая проходила в Санта Барбаре, США. Данный метод позволяет злоумышленникам получать секретные AES ключи в пять раз быстрее, чем это делалось ранее. Он же использует технику двудольного криптоанализа для удаления двух битов из 128-, 192- и 256-битных ключей. И эксперты поспешили успокоить общественность: «Данное исследование пошатнуло многие основы, так как мы получили первый метод взлома, AES-шифра с одним ключом, (немного) более быстрый, чем брут форс атака, – заявил Нейт Лаусон, криптограф и начальник отдела консультаций по безопасности компании Root Labs. – Однако с практической точки зрения, AES-шифр взломать пока невозможно» [7].

Тем не менее уже меньше, чем через год, Сергей Скоробогатов, который до этого недавно обнаружил бэкдор в китайском чипе FPGA, произвел успешное извлечение ключа AES, который отмечен как «высокозащищенный» и «практически неломаемый», на FPGA военного уровня Actel/Microsemi ProASIC3 за 0,01 секунду. Использовался способ анализа сторонних каналов, который называется Pipe-

line Emission Analysis (PEA). Это не новый способ сам по себе, а значительно улучшенный метод волнового анализа. Данный способ используется для атаки на реализацию AES на FPGA, используя наводки и помехи [8].

Вывод: данный алгоритм оказался тоже взламываемым при помощи метода волнового анализа. Также существует вероятность, что при помощи квантовых компьютеров в будущем это будет сделать достаточно быстро, чем это предполагали, когда создавали алгоритм.

3. **RSA** (аббревиатура от фамилий Rivest, Shamir и Adleman) – криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.

Криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи. Алгоритм используется в большом числе криптографических приложений, включая PGP, S/MIME, TLS/SSL, IPSEC/IKE и других [2].

Основная проблема. Была обнаружена уязвимость в реализации алгоритма RSA, позволившая исследователям взломать 1024-битное шифрование. Но для реализации взлома необходим физический доступ к держателю «секретного ключа», так что на крупных компаниях это вряд ли отразится.

Что же касается обычных гаджетов, то достаточно не упускать их из виду. Докторант Мичиганского университета Андреа Пеллегрини (Andrea Pellegrini) завтра представит отчет о продланной работе на конференции «Design, Automation and Test in Europe» (DATE) в Дрездене.

Но перейдем к сути. Атака на алгоритм проводилась путем искусственного вызова ошибок при помощи изменения напряжения на процессоре. В результате появлялись ошибки в коммуникации с другими клиентами, и удавалось получить небольшую часть ключа, а как только было собрано достаточно частей, ключ был восстановлен в режиме офлайн. На все про все ушло 104 часа работы 81 процессора Pentium 4. Техника не пострадала, следов взлома не осталось. Несмотря на то, что в статье описывается только уязвимость, ученые из университета заявили, что они предлагают довольно простое решение проблемы. Для этого, по их словам, достаточно применить «соль», позволяющую изменять порядок цифр случайным образом при каждом запросе ключа [9].

Таким образом, RSA также был взломан, а алгоритм до сих пор не изменен, следовательно, если бы ученые стремились заново его взломать, то потратили даже меньше 104 часов, потому что вычислительная мощность компьютеров увеличилась.

Заключение

Распространенность алгоритмов шифрования часто служит им плохую службу, потому что все больше злоумышленников и не только пытаются их взломать. Однако стоит отметить, что для своего времени каждый был прорывом в шифровании. И новое время требует новое откровение в среде шифрования данных.

ЛИТЕРАТУРА

7. The 9 Most In-Demand Programming Languages of 2016 [Electronic resource]. – 2016. – Mode of access: <http://www.codingdojo.com/blog/9-most-in-demand-programming-languages-of-2016/>. – Date of access: 23.09.2017.
8. RSA [Электронный ресурс]. – 2017. – Режим доступа: <https://ru.wikipedia.org/wiki/RSA/>. – Дата доступа: 23.09.2017.
9. AdvancedEncryptionStandard [Электронный ресурс]. – 2017. – Режим доступа: https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard. – Дата доступа: 23.09.2017.
10. DES [Электронный источник]. – 2017. – Режим доступа: <https://ru.wikipedia.org/wiki/DES>. – Дата доступа: 23.09.2017.
11. Криптографические основы безопасности [Электронный ресурс]. – 2017. – Режим доступа: <http://www.intuit.ru/studies/courses/28/28/lecture/20412?page=4#sect17>. – Дата доступа: 23.09.2017.
12. Алгоритм шифрования AES [Электронный ресурс]. – 2017. – Режим доступа: <https://www.opengsm.ru/blog/algorithm-shifrovaniya-aes/>. – Дата доступа: 23.09.2017.
13. Криптографы нашли способ взлома ключей AES-шифра [Электронный ресурс]. – 2011. – Режим доступа: <http://www.securitylab.ru/news/406880.php>. – Дата доступа: 23.09.2017.
14. Извлечение AES ключа в мгновение ока [Электронный ресурс]. – 2012. – Режим доступа: <https://habrahabr.ru/post/145775/>. – Дата доступа: 23.09.2017.
15. Взломано 1024-битное шифрование RSA [Электронный источник]. – 2010. – Режим доступа: <https://habrahabr.ru/post/86841/>. – Дата доступа: 23.09.2017.

УДК 004.02

ОБЗОР СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ, ОСНОВАННЫХ НА MYSQL

В.С. КУПЦОВ

(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Рассматриваются плюсы и минусы систем управления базами данных, основанных на MySQL. Проанализированы возможности самых популярных форков. Рассмотрены новые хранилища данных.

Сервер реляционной базы данных MySQL в короткое время успел стать сверхпопулярной базой данных, а также незаменимой частью современного Интернета, входя в связку открытых веб-технологий LAMP (Linux-Apache-MySQL-PHP), которая и формирует технологически практически весь современный Веб. Роль баз данных в этой связке имеет исключительный характер. Поэтому архитектура современного сайта всегда содержит быстрое и гибкое хранилище данных, такое как MySQL. Также стоит отметить большое количество систем управления базами данных, основанных на MySQL.

Далее будут рассмотрены возможности самых популярных форков (ответвлений кода от основного проекта) MySQL.

ExtSQ – проект параллельно ведет две ветви разработки, базирующихся соответственно на кодовой ветке 5-й версии оригинального MySQL. ExtSQL разрабатывался с сильным уклоном для специализированного использования в системах веб-хостинга и призван решить проблемы, связанные с организацией учета потребления ресурсов. Администраторы ExtSQL получают возможность полного мониторинга активности пользователей, всех баз и соединений. Например, запрос «SHOW STATISTICS select, insert FROM user HISTORY» позволит узнать число запросов «select» и «insert», совершенных пользователями за последний час. Естественно, для этого в бывший MySQL добавлены новые команды и расширен диалект SQL, при этом сохранена полная обратная совместимость с MySQL. Плата за подобные надстройки над традиционным MySQL выражается в потере производительности сервера в среднем на 5–6% на обычных задачах, и на 12–16% при интенсивной работе с реально большими базами данных.

PerconaServer – сборка MySQL с включенным по умолчанию XtraDBstorageengine. Отличается от MySQL+InnoDBplugin лучшей производительностью и масштабируемостью, особенно на современных многоядерных серверах. Также улучшена функциональность – больше полезной для оптимизации статистики и пр. Полностью совместим с таблицами InnoDB, то есть можно переходить от InnoDB к XtraDB и обратно без проблем. Хранилище XtraDB основано на коде InnoDB-plugin, полностью совместимо с ним, но отличается заметно более высокой производительностью, благодаря интеграции патчей от компаний Google и Percona. В частности, в XtraDB улучшен механизм работы с памятью, улучшена работа подсистемы ввода/вывода InnoDB, добавлена поддержка нескольких потоков чтения и записи, поддержка управления пропускной способностью, реализация упреждающей выборки данных (read-ahead), адаптивная установка контрольных точек (adaptivecheckpointing), расширены возможности по масштабированию для больших проектов, система организации блокировок адаптирована для работы на системах с большим числом CPU, добавлены дополнительные возможности для накопления и анализа статистик.

MariaDB – сборка, синхронизированная с кодовой базой MySQL и полностью с ней совместимая, то есть может выступать в качестве прозрачной замены MySQL, обладая при этом рядом расширенных функций, включая оптимизации производительности и поставляясь с набором дополнительных движков хранилищ.

Новые хранилища данных:

- 1) **Aria** – высоконадежное хранилище, отличающееся повышенной устойчивостью сохранения целостности данных после краха, при полной совместимости с MyISAM;
- 2) **OQGRAPH** – хранилище для организации сложных графов;
- 3) **Sphinx** – хранилище для построения поисковых движков, обеспечивает высокую производительность индексации и поиска, позволяя делать запросы с помощью SphinxQL. Хорошо масштабируется, позволяя работать с миллиардами документов с возможностью распределенного поиска;
- 4) **PBXT (PrimeBase XT)**, поддерживающее мультиверсионный метод организации хранения данных MVCC (multi-versionconcurrencycontrol), позволяющий избавиться от блокировок при выполнении операций чтения. Поддерживает ACID-совместимые транзакции, быстрый откат транзакций и восстановление после некорректного завершения работы сервера. Имеются средства для обеспечения ссылочной целостности данных, поддержка определения внешних ключей, каскадных обновлений и удалений данных;
- 5) **FederatedX** – позволяет организовать обращение к удаленным таблицам как к локальным.

Другие улучшения:

- 1) патчи MyISAM движка – Сегментированный кэш, который при высоких нагрузках дает существенный прирост производительности;
- 2) виртуальные столбцы;
- 3) ликвидация таблиц – новый вид оптимизации запросов с использованием JOIN;
- 4) пул потоков – теперь на одно соединение можно открыть больше одного потока;
- 5) улучшены механизмы отладки медленных запросов.

Совершенно противоположным путем пошел другой, уже достаточно популярный и перспективный форк **MySQL** – проект Drizzle. Этот проект представляет собой упрощенный и более быстрый вариант MySQL, в котором тщательно отобраны и удалены все ресурсоемкие и маловостребованные возможности MySQL 5. Часть из этих возможностей все же возможно реализовать через подключаемые плагины. Эта СУБД позиционируется как высокоскоростная и высоконадежная БД, с поддержкой ACID-транзакций. В качестве хранилища используется InnoDB и PBXT. Исходный код MySQL, написанный на С, был полностью переписан на языке С++. Управление проектом находится в руках независимого сообщества. Архитектура Drizzle построена на максимальном упрощении структуры БД и выносе логики на сторону приложений. В частности, такой дизайн СУБД позволяет организовать обработку большого числа параллельных запросов, при выполнении которых в полной мере задействуются мощности современных многоядерных CPU. Кроме этого, в Drizzle дополнительно реализованы встроенные средства для разнесения данных по ключевому полю (sharding) на кластер из нескольких машин, для создания эффективной балансировки нагрузки для по-настоящему сверхнагруженных проектов. По сравнению с MySQL в Drizzle удалена поддержка хранимых процедур (вместо CREATE FUNCTION следует использовать связываемые объекты), триггеров, кэша запросов (querycache), представлений (view), операции GRANT и ALTER, ограничений ACL, команды SHOW, предварительно подготовленных запросов (preparedstatement) и др. Прекращена поддержка малоиспользуемых типов данных из MySQL.

OurDelta – это ответвление от MySQL отбирает самые интересные доработки от независимых разработчиков, после чего собирает, интегрирует и синхронизирует их всех в одном продукте. Чтобы дать возможность выбора, OurDelta ведет две параллельные ветви своих популярных ребилдов: на базе исходников классического MySQL и вышерассмотренного MariaDB. Чтобы иметь представление, что же включается в билды от OurDelta, остановимся на одном из ее крупных разработчиков патчей – проекте Facebook. Чтобы справиться с нагрузкой, которую создают более миллиарда пользователей социальной сети, Facebook оперирует четырьмя тысячами экземпляров MySQL (при этом очень активно используется шардинг), всего задействовано девять тысяч инсталляций memcached. Конечно, стандартный MySQL не предназначен для подобных нагрузок, поэтому уже более 5 лет идет интенсивная доработка MySQL под собственные экстремальные потребности Facebook. Неудивительно, что patchset от Facebook – один из самых интересных в составе OurDelta.

В заключение можно сделать **вывод**, что выбирать систему управления базами данных следует в зависимости от размеров, потребностей проекта и нагрузки на нем.

ЛИТЕРАТУРА

1. Bloggerator'snewslne – новостной HTTP-блог об ИТ [Электронный ресурс] / MySQL«на стероидах» – Режим доступа: <http://bloggerator.org/page/mysql-na-steroidah>. – Дата доступа: 20.09.2017.
2. Хабрахабр [Электронный ресурс] / Форки движка MySQL: MariaDB, Percona. whoiswho? – Режим доступа: <https://habrahabr.ru/post/108104/>. – Дата доступа: 20.09.2017.

УДК 004.02

**ERP-СИСТЕМА КАК ИНСТРУМЕНТ ДЛЯ СТРУКТУРИРОВАНИЯ
И АВТОМАТИЗАЦИИ ПРОЦЕССА УПРАВЛЕНИЯ ПРЕДПРИЯТИЕМ****В.С. КУПЦОВ***(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматривается понятие системы управления предприятием (ERP-системы) и цели их внедрения в работу предприятия. Показано, что внедрение системы управления предприятием делает бизнес-процессы компании более прозрачными, повышая ее привлекательность для инвесторов. Использование данных систем целесообразно и позволяет упростить и оптимизировать работу предприятия.

ERP-системы (EnterpriseResourcePlanning, или Планирование Ресурсов Предприятия) являются удобным инструментом для современных руководителей. Системы класса ERP – это многофункциональный набор связанных между собой приложений в виде среды для автоматизации планирования, контроля, анализа и управления бизнес-процессами предприятия в сфере финансов, товарно-материальных, производственных и логистических потоков, бухгалтерского учета, управления персоналом, отношений с контрагентами.

К наиболее популярным ERP-системам на территории СНГ относятся:

- 1) 1C:ERP;
- 2) MS Dynamics ERP;
- 3) Sage ERP X3;
- 4) Ultima;
- 5) Инталев 24;
- 6) NetSuite;
- 7) КОМПАС;
- 8) Oracle E-Business Suite и др.

Системы управления ресурсами компании внедряются для того, чтобы объединить все подразделения компании и все необходимые функции в одной компьютерной системе, которая будет обслуживать текущие потребности этих подразделений. Тогда как традиционные системы используют многоступенчатую схему обработки данных, при которой за фиксацией информации на бумаге следует передача бумажных носителей в конкретное структурное подразделение, где данные переформируются или вводятся в компьютер, в системах управления предприятием помещение информации в базу данных производится непосредственно от источника, минуя промежуточные стадии, после чего данные становятся доступны всем заинтересованным лицам. Система ведет единую базу данных по всем подразделениям и задачам, так что доступ к информации становится проще, а главное, подразделения получают возможность обмениваться информацией.

ERP-система автоматизирует задачи, встроенные в выполнение бизнес-процессов. Так, при получении заказа от потребителя менеджер имеет всю информацию об отношениях с заказчиком. Когда одно подразделение заканчивает работу с заказом, тот автоматически передается в следующее подразделение. При этом исключается возможность возникновения информационных несоответствий внутри системы, что значительно повышает качество информации и обеспечивает дополнительные преимущества при принятии решений. В итоге заказы обрабатываются быстрее и без ошибок. Аналогичные возможности возникают у многих других служб – службы персонала, производственного отдела, отдела маркетинга, службы снабжения. Единая информационная база позволяет учитывать взаимосвязь отдельных процессов, как, например, загрузка заказами на текущий месяц и график отпусков персонала. Благодаря широкой доступности информации отпадает необходимость в посреднических услугах по передаче данных между сотрудниками или от руководства компании к ее служащим. Данное обстоятельство, в свою очередь, позволяет сократить штат компании за счет работников, чьи функциональные обязанности ограничиваются исключительно подготовкой и распространением информации.

Система меняет отношение персонала в отдельности и служб в целом к своей работе. Теперь менеджеры, ведущие заказ, владеют информацией о его состоянии в каждый момент времени и по всем аспектам: поступила ли оплата за работу, достаточно ли комплектующих на складе, стоит ли в графике выполнения работ их заказ к нужному сроку. И если, например, склад вовремя не введет в систему информацию о запасах, то получивший запрос клиента менеджер, справившись в системе о состоянии

склада, может ответить отказом на его обращение, увидев, что запас на складе недостаточен или нужное изделие отсутствует. Тем самым система повышает ответственность каждого за общее дело.

Благодаря единовременному вводу данных в общую базу предприятия информация может использоваться одновременно любым количеством пользователей системы, что повышает эффективность как планирования, так и контроля, а также обеспечивает согласованность действий всех звеньев предприятия и продуктивность управления в целом.

ERP-системы с веб-интерфейсом делают возможным объединение географически удаленных друг от друга филиалов и отделений компании и вовлечение их в единое информационное пространство, что позволяет осуществлять обмен организационными наработками между структурными подразделениями с нестандартными и высокоэффективными процессами, а также способствует формированию единого корпоративного образа и улучшению имиджа компании на мировом рынке.

Системы управления предприятием позволяют создавать своего рода информационные каналы, способные связать предприятие с компаниями-партнерами для предоставления им доступа к базе данных предприятия и обмена необходимой информацией. В результате такого сотрудничества формируется единый информационный архив, что позволяет оптимизировать процессы снабжения, поставки и другие виды совместной деятельности, а также вывести на качественно новый уровень весь процесс взаимодействия в целом.

Внедрение системы управлением предприятия делает бизнес-процессы компании более прозрачными, повышая ее привлекательность для инвесторов, а следовательно, позволяет привлечь дополнительные инвестиции для дальнейшего развития производства.

Внедрение такой системы, как правило, является длительным и сложным процессом. У каждого крупного производителя ERP системы разработаны свои технологии и методики внедрения. Эти методики несколько отличаются друг от друга, но общий порядок действий, в основном, одинаковый. Кроме того, порядок внедрения может различаться в зависимости от программной и аппаратной архитектуры системы. На эти методики стоит обращать внимание, когда осуществляется выбор поставщика услуг.

Организации, которая приняла решение внедрять ERP систему, необходимо со своей стороны предпринять ряд действий для успешной реализации проекта внедрения.

К таким действиям относятся:

1. *Подготовка внедрения.* На этапе подготовки организации необходимо определить основные цели, ради чего будет внедряться ERP система. Это позволит четко понять результаты и ожидания от внедрения. Так как внедрение системы является довольно затратным проектом, необходимо оценить предполагаемый бюджет, который организация сможет выделить на внедрение. Также, на этом этапе определяется ответственный за проект, и выявляются ключевые специалисты (члены команды проекта со стороны организации), определяется порядок их взаимодействия.

2. *Анализ бизнеса.* Эти действия являются одними из самых критических во всем проекте. Как правило, ERP система должна работать в течение десятка лет и более. Поэтому необходимо определить перспективы развития как самой организации, так и рынка на длительный период.

3. *Выбор системы управления ресурсами предприятия.* Выбор системы необходимо осуществлять на основе текущих и будущих потребностей бизнеса.

4. *Выбор поставщика.* Как правило, на рынке существует много поставщиков одной и той же ERP системы. Это компании вендоры или системные интеграторы. Они предлагают комплекс услуг по техническому внедрению системы.

5. *Управление проектом.* Ответственный за внедрение ERP системы и специалисты организации должны осуществлять управление проектом внутри компании. Они должны вести план проекта, контролировать сроки, бюджет, объем работ, соответствие целям внедрения.

6. *Тестирование.* Даже при самой лучшей организации внедрения, существует риск возникновения ошибок в работе системы. Поэтому по ходу ввода в эксплуатацию функционала ERP системы необходимо предусмотреть обязательное тестирование работы процессов, подразделений и модулей. Лучшим способом тестирования является выполнение параллельной работы в старой системе и во внедряемой системе. Это позволит избавиться от основных ошибок.

7. *Тренинги и обучение.* Обучение сотрудников является обязательным условием работы системы. В зависимости от сложности системы оно может занять несколько недель. Перед началом обучения необходимо убедиться, что ERP система была протестирована. Одним из неудачных вариантов является случай, когда поставщик услуг совмещает тестирование и обучение пользователей. В этом варианте сотрудники организации (будущие пользователи системы) выступают в качестве тестировщиков системы.

8. *Ввод в эксплуатацию.* Ввод в эксплуатацию является важным моментом в процессе внедрения ERP системы. Существует два варианта ввода системы в эксплуатацию:

- 1) запуск системы сразу во всей организации;
- 2) поэтапный ввод.

Второй вариант является более предпочтительным, так как позволяет постепенно переходить на новые условия работы. В случае возникновения ошибок или проблем в работе, будет затронута только часть бизнеса (отдельные процессы или подразделения), а не вся организация.

В заключение можно сделать *вывод*, что, несмотря на довольно высокую стоимость ERP-систем и некоторые сложности при внедрении, использование данных систем целесообразно и позволяет упростить и оптимизировать работу предприятия.

ЛИТЕРАТУРА

1. ЦМД-софт [Электронный ресурс] / ERP | Что такое ERP-система? – Режим доступа: http://www.cmdsoft.ru/information_systems/erp/what_is_erp_system/. – Дата доступа: 20.09.2017.
2. Онлайн бухгалтерия и ERP [Электронный ресурс] / Топ 10: Онлайн ERP системы. – Режим доступа: <http://www.clouderp.ru/tools/erp/>. – Дата доступа 20.09.2017.
3. Хабрахабр [Электронный ресурс] / Что такое ERP система. – Режим доступа: <https://habrahabr.ru/post/333018/>. – Дата доступа: 20.09.2017.

УДК 004.042

ПРОБЛЕМЫ ПАРАЛЛЕЛИЗМА В ПРОГРАММИРОВАНИИ

Ю.В. ЛАПТЕВ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматриваются некоторые принципиальные проблемы обеспечения параллельной работы программных приложений и способы их решения.

Параллельное выполнение операций – одна из наиболее сложных дисциплин в области разработки программного обеспечения. Проблемы параллелизма возникают каждый раз, когда несколько процессов или потоков вычислений предпринимают попытки манипуляций одними и теми же данными. Также параллельные операции трудно тестировать и невозможно перечислить все возможные сценарии развития событий, способным привести к тем или иным неприятным последствиям [1].

Проблемы параллелизма

Рассмотрим некоторые проблемы обеспечения параллельной работы приложений, на преодоление которых направлены усилия систем управления параллельными заданиями.

Рассмотрим *пример утраты изменений*. Предположим, что разработчик А открывает на редактирование файл Customer, с исходным кодом, и правит в нем какой-то метод. В это же время разработчик В обращается к этому же файлу и правит тот же метод. Делает он это быстрее первого разработчика и сохраняет изменения. Разработчик А об этом ничего не знает, и как итог, сохраняя свои изменения, перезапишет все изменения второго разработчика. Данная проблема выражается в потере достоверности информации и в некорректном поведении системы, чего можно было бы избежать, если бы два человека не обращались к одним и тем же данным одновременно. Естественно можно упорядочить действия таким образом, чтобы только один субъект имел право обращаться к данным в определенный момент времени, но это снизит возможность параллельной обработки информации.

Также имеется одна из весьма важных и трудных проблем, связанная с взаимоблокировками. Предположим, что разработчик А приступает к редактированию файла Customer, а разработчик В редактирует файл Order. В какой-то момент времени разработчик В понимает, что для завершения работы ему необходимо несколько изменить и файл Customer, но разработчик А владеет блокировкой на этот файл и второму разработчику приходится ждать ее освобождения. В то же время первому разработчику приходит мысль, что нужно подправить файл Order, но этому мешает блокировка, удерживаемая разработчиком В. Оба разработчика попадают в ситуацию взаимоблокировок, и ни один не может завершить свою работу, пока другой разработчик не завершит свою. В данном случае проблема не выглядит угрожающе и ее несложно предотвратить, но если цепочку взаимозависимостей составляет множество людей или программ, тогда и начинаются серьезные проблемы.

Рассмотрим еще одну проблему с бесконечной отсрочкой, когда одна или несколько задач должны ожидать до тех пор, пока не произойдет какое-то событие или не создадутся определенные условия. Предположим, что если установить для электронного банка правила, предписывающие всем задачам снятия денег со счета находиться в состоянии ожидания до тех пор, пока не будут выполнены все задачи вложения денег на счет, то задачи снятия денег рискуют стать бесконечно отсроченными. Если будут без конца поступать запросы на пополнение одного и того же счета, то такая ситуация может вызвать бесконечную отсрочку задач снятия денег [2].

Транзакции

Транзакции являются основным инструментом управления параллельными процессами в приложениях. Транзакция представляет собой ограниченную последовательность действий с явно определенными начальной и завершающей операциями. Например, транзакция, связанная с банкоматом, начинается с размещения магнитной карты в считывающем устройстве и завершается выдачей денег или сообщением о несоответствии запрошенной суммы остатку на счете. Все ресурсы, затрагиваемые транзакцией, пребывают в согласованном состоянии в момент ее начала и остаются в таковом после ее завершения. Кроме этого, транзакция должна либо выполняться целиком, либо не выполняться вовсе. Банковская система не может вычесть сумму остатка на счете до тех пор, пока клиенту не будет выдана соответствующая пачка купюр. Транзакции часто описывают в терминах свойств, которые обозначаются аббревиатурой ACID:

- **atomicity** (атомарность). В контексте транзакции либо выполняются все действия, либо не выполняется ни одно из них. Частичное или избирательное выполнение недопустимо. Например, если клиент банка переводит сумму с одного счета на другой и в момент между завершением расходной и началом приходной операции сервер терпит крах, система должна вести себя так, будто расходной операции

не было вовсе. Система должна либо осуществить обе операции, либо не выполнить ни одной. Фиксация результатов служит свидетельством успешного окончания транзакции, а откат приводит систему в состояние, в котором она пребывала до начала транзакции;

- **consistency** (согласованность). Системные ресурсы должны пребывать в целостном и непротиворечивом состоянии как до начала транзакции, так и после ее окончания;

- **isolation** (изолированность). Промежуточные результаты транзакции должны быть закрыты для доступа со стороны любой другой действующей транзакции до момента их фиксации. Иными словами, транзакция протекает так, будто в тот же период времени других параллельных транзакций не существует;

- **durability** (устойчивость). Результат выполнения завершенной транзакции не должен быть утрачен ни при каких условиях.

В большинстве случаев под транзакциями в приложениях подразумеваются последовательности операций, описывающие процессы взаимодействия с базами данных. Но существует и множество других объектов, управляемых с помощью механизмов поддержки транзакций: например, очереди сообщений или заданий на печать, банкоматы и т.д. Для обеспечения высокого уровня пропускной способности современные системы управления транзакциями проектируются в расчете на максимально короткие транзакции. Обычно из практики исключаются транзакции, которые охватывают действия по обработке нескольких запросов. Если же подобной ситуации избежать не удастся, решения реализуются на основе схемы длинных транзакций. Чаше границы транзакции совпадают с моментами начала и завершения обработки одного запроса. Подобная транзакция запроса – весьма удачная модель, и многие среды поддерживают простой и естественный синтаксис ее описания. Альтернативное решение состоит в том, чтобы как можно дольше откладывать процедуру открытия транзакции. При использовании подобной отсроченной транзакции все операции чтения могут выполняться до момента ее начала, который наступает только при необходимости осуществления операций, связанных с внесением каких бы то ни было изменений. Это позволяет минимизировать длину транзакции, но лишает возможности на протяжении продолжительных периодов времени применять какие-либо средства управления параллельными операциями. Такая стратегия сопряжена с повышением вероятности возникновения эффекта несогласованного чтения и потому используется сравнительно редко [3].

Применяя транзакции, следует понимать, что именно надлежит блокировать. Во многих случаях диспетчер транзакций СУБД блокирует отдельные записи таблицы базы данных, вовлеченные в операцию, что позволяет сохранить высокий уровень параллелизма при доступе к таблице. Но если транзакция пытается блокировать слишком большое количество записей таблицы, число запросов на блокировку превышает допустимый лимит и система распространяет действие блокировки на таблицу в целом, приостанавливая выполнение конкурирующих транзакций.

Заключение

Параллелизм в программировании – достаточно обширная и сложная тема. Модель транзакций позволяет избежать массы трудностей, но это не значит, что проблемами управления параллельными заданиями можно полностью пренебречь, так как многие аспекты взаимодействия приложения и системы нельзя свести к контексту единой транзакции. Во многих случаях приходится иметь дело с данными, модифицируемыми несколькими транзакциями. К тому же существуют более сложные системы с поддержкой множества одновременно протекающих потоков вычислений, где приходится применять более сложные подходы к решению проблем параллелизма.

ЛИТЕРАТУРА

1. Параллелизм [Электронный источник]. – 2017. – Режим доступа: [https://ru.wikipedia.org/wiki/Параллелизм_\(информатика\)](https://ru.wikipedia.org/wiki/Параллелизм_(информатика)). – Дата доступа: 27.09.2017.
2. Проблемы параллельного и распределенного программирования [Электронный источник]. – 2017. – Режим доступа: http://www.k2x2.info/kompyutery_i_internet/parallelnoe_i_raspredelennoe_programmirovanie_na_s/p4.php. – Дата доступа: 27.09.2017.
3. Фаулер, М. Архитектура корпоративных программных приложений : пер. с англ. / Мартин Фаулер. – М. : Издат. дом «Вильямс», 2006.

УДК 004.62

ПРИНЦИПЫ И ПРИЕМЫ ОБРАБОТКИ ОЧЕРЕДЕЙ

Ю.В. ЛАПТЕВ*(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)*

Рассматривается понятие очередей, их применение в высоконагруженных проектах и принципы обработки. Показаны области применения очередей, инфраструктура обработки очередей.

Очередью в программировании называется упорядоченный набор элементов, которые могут удаляться с ее начала и помещаться в ее конец. Очередь в программировании используется, как и в реальной жизни, когда нужно последовательно совершить какие-то действия в порядке их поступления. Примером может служить организация событий в Windows. Когда пользователь оказывает какое-то действие на приложение, то в приложении не вызывается соответствующая процедура (в этот момент приложение может совершать другие действия), а ему присылается сообщение, содержащее информацию о совершенном действии, это сообщение ставится в очередь, и только когда будут обработаны сообщения, пришедшие ранее, приложение выполнит необходимое действие [1].

Области применения очередей

Самый распространенный способ применения – для асинхронного выполнения операций, порождаемых в процессе обработки запросов пользователя, результаты исполнения которых не требуются немедленно для осуществления обратной связи. Например, отправка почты и уведомлений, предложений дружбы, обновление поисковых индексов, сброс кэшей, постобработка данных, полученных от пользователя (перекодирование видео, нарезка фотографий), отправка данных подсистемам, время реакции/надежность которых не гарантируется. Обладая удобной инфраструктурой для распределенной обработки заданий, спектр применения очередей является еще больше:

- *отложенная обработка действий пользователя.* Не лучшая идея заставлять пользователя ждать, пока будут сохранены его данные в SQL базу или другое хранилище (а зачастую необходимо вносить изменения сразу в несколько систем). Иногда данные могут вовсе не попасть в хранилище, если клиент оборвал соединение, не дождавись ответа. Хорошей практикой является добавление события о пользовательских действиях в достаточно быстрый сервер очередей, после чего можно отвечать клиенту, что операция прошла успешно. Всю остальную работу надежно и эффективно выполнят обработчики очереди;

- *рассылка сообщений, писем и похожие операции.* Если необходимо отослать большое количество данных, при этом не перегрузив хранилища всплеском запросов, то это делается с помощью варьирования количества обработчиков очереди, размазывая пиковую нагрузку до разумного уровня, чтобы время обработки клиентских запросов не ухудшилось. И самое главное – с помощью очередей легко избежать дублирования сообщений. Крайне неприятно получить два письма об одном и том же событии;

- *транспорт для «надежной» статистики.* Передача важных (все, что связано с деньгами) данных на агрегаторы статистики. Системы агрегации статистики обычно требовательны к ресурсам процессора, и при обработке данных могут не обеспечивать необходимое для frontend-серверов время ответа. Еще одна особенность подобных систем – неравномерная загрузка, обычно связанная с обработкой данных порциями. Передача статистики через серверы очередей позволит избежать проблем с нестабильной задержкой и при этом сохранит гарантию доставки;

- *группировка событий.* Если группа событий будет обращаться к одному и тому же набору данных в других системах, имеет смысл ставить одинаковое время активации. Физический смысл ухищрений в более эффективном использовании кэшей хранилища, в которое пойдут запросы из обработчиков сообщений;

- *каскадные очереди.* Организация конечного автомата из нескольких очередей путем переключения данных между очередями по завершении очередного этапа обработки. Часто необходимо в процессе обработки сообщения выполнить ряд действий, сильно различающихся по необходимым ресурсам. В таком случае разнесение «быстрых» и «медленных» действий по разным этапам позволяет эффективно управлять необходимым количеством ресурсов, варьируя число обработчиков для каждой очереди [2].

Инфраструктура обработки очередей

По мере увеличения количества типов очередей уследить за возрастающим количеством скриптов и демонов (срор) становится все сложнее. Нужно следить за нагрузкой, запускать достаточное количество процессов, не забывать мониторить их, а при выпадении серверов своевременно поднимать процессы на других машинах. Для решения всех этих проблем нужна инфраструктура, которой нужно предоставить

кластер серверов-обработчиков и серверы очередей, а она сама уже будет управлять всем этим. При проектировании инфраструктуры для обработки очередей необходимо учитывать следующие характеристики:

- **распределенность** – задания одного типа могут выполняться на разных серверах;
 - **отказоустойчивость** – выпадение 10-20 процентов серверов кластера не должно приводить к деградации важных очередей;
 - **гомогенность** – отсутствие какой-либо специализации серверов, из-за чего любое задание может выполняться на любом сервере;
 - **приоритизация** – есть критичные очереди, а есть те, которые могут и подождать;
 - **автоматизация** – минимум ручного вмешательства: балансировка – перераспределение обработчиков каждого типа в зависимости от потребностей; адаптация к серверу – определение количества обработчиков на каждом сервере в зависимости от его характеристик, отслеживание потребления памяти и процессора;
 - **пакетная обработка** – задания одного типа должны группироваться для возможности применения оптимизаций;
 - **статистика** – сбор данных о поведении очередей, потоке заданий, количестве, наличии ошибок;
 - **мониторинг** – оповещение о нештатных ситуациях и изменении стандартного поведения очередей.
- Схема инфраструктуры обработки очередей представлена на рисунке 1.

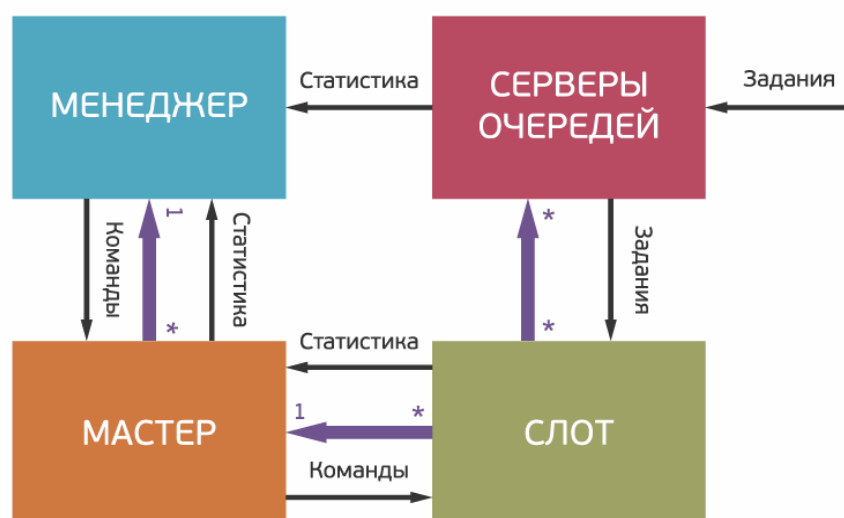


Рисунок 1. – Схема инфраструктуры обработки очередей

На данной схеме можно выделить 3 базовые сущности: слот, мастер и менеджер.

Слот – это процесс-обработчик очереди (воркер). Слотов на каждом сервере может быть запущено множество, каждый слот характеризуется типом очереди, которую он в данный момент обрабатывает. Специализация слотов на очередях определенных типов повышает локальность соединений к базам данных и внешним сервисам, а также упрощает отладку.

Мастер представляет собою процесс, ответственный за работу слотов на определенном сервере. В его обязанности входит запуск и остановка слотов, поддержание оптимального количества слотов на сервере в зависимости от его конфигурации, доступной памяти, а также мониторинг жизнедеятельности слотов, уничтожение зависших и использовавших слишком много системных ресурсов слотов.

Сущность *менеджер* выступает в роли процесса-руководителя кластера, который запускается в одном экземпляре и отвечает за весь кластер. Данный процесс на основе данных от сервера очередей и статистических данных от мастеров определяет необходимое количество слотов каждого типа и решает, какой конкретный слот будет заниматься заданиями какого типа [3].

ЛИТЕРАТУРА

1. Структуры данных, Очередь [Электронный источник]. – 2017. – Режим доступа: <https://prog-cpp.ru/data-queue/>. – Дата доступа: 27.09.2017.
2. Сервер очередей [Электронный источник]. – 2017. – Режим доступа: <https://habrahabr.ru/company/mailru/blog/216363/>. – Дата доступа: 27.09.2017.
3. Инфраструктура обработки очередей с социальной сети Мой Мир [Электронный источник]. – 2017. – Режим доступа: <https://habrahabr.ru/company/mailru/blog/228131/>. – Дата доступа: 27.09.2017.

УДК 004.021

ПРОЕКТИРОВАНИЕ REALTIME ВЕБ-ПРИЛОЖЕНИЙ ПРИ ПОМОЩИ WEBSOCKET И СЕРВЕРНЫХ СОБЫТИЙ

А.А. ЛЕВИЦКИЙ

(Представлено: Т.С. РУДЬКОВА)

Представлено теоретическое объяснение работы WebSocket-протокола, а также практический способ создания realtime веб-приложений при помощи данной технологии. Также содержится историческая справка о становлении данной технологии стандартом.

Когда данные на сервере меняются, пользователям необходимо дать об этом знать без необходимости действий с их стороны. Это один из видов увеличения производительности, который избавляет пользователя от однотипных действий, таких как обновление страницы.

Ранее приходилось эмулировать серверные события, используя для этого метод long-polling, заставляющий пользователей делать «длинные» запросы, которые оставались открытыми до того, как сервер будет готов отправить какое-либо сообщение в ответ. После получения сообщения, запрос будет закрыт, а новый – сформирован. Другое решение этой проблемы – <iframe> хаки и Flash, но данные методы не идеальны, так как требуют реализации нестандартного поведения технологий, которые не предназначены для задач, связанных с серверными событиями, а также неудобны в использовании.

Принцип работы WebSocket и алгоритмы его реализации. В 2006 году компания Opera представила серверные события (SSE) из спецификаций WHATWG Web Applications 1.0. Эта технология позволяла потоково передавать события непрерывно из веб-сервера прямо в клиентский браузер. Другие браузеры последовали этому примеру и стали реализовывать этот стандарт в 2011 году как часть спецификаций HTML5. В этом же году WebSocket-протокол был стандартизирован [2].

WebSocket позволяет открыть двустороннее соединение между клиентом и сервером, предоставляя возможность пересылать сообщения клиенту, когда какие-либо данные на сервере меняются, без необходимости клиента запрашивать эти самые данные. Это самая важная вещь для отзывчивости приложения с большим количеством одновременных соединений, например, онлайн-игры или мессенджеры. Однако до появления библиотеки socket.io, вышедшей в 2014 году, это было затруднительно [1]. Впоследствии эта библиотека стала стандартом использования WebSocket в современных веб-приложениях.

Конструктор WebSocket инициализирует соединение с сервером через протокол ws или wss (secure – безопасный). Этот протокол поддерживают все современные браузеры как на компьютерах, так и на смартфонах. Таблица поддержки браузеров проиллюстрирована на рисунке 1.

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
			59						
			60	10.1		10.2		4.4	
	14	54				10.3		4.4.4	
11	15	55	61	11	47	11	all	56	61
	16	56	62	TP	48				
		57	63		49				
		58	64						

Рисунок 1. – Поддержка браузеров WebSocket

Однако такая технология, как WebSocket, требует безупречной отладки, что невозможно достичь без специальных инструментов. Например, Google Chrome предоставляет возможность наблюдать сообщения при помощи вкладки Frame (рисунок 2).

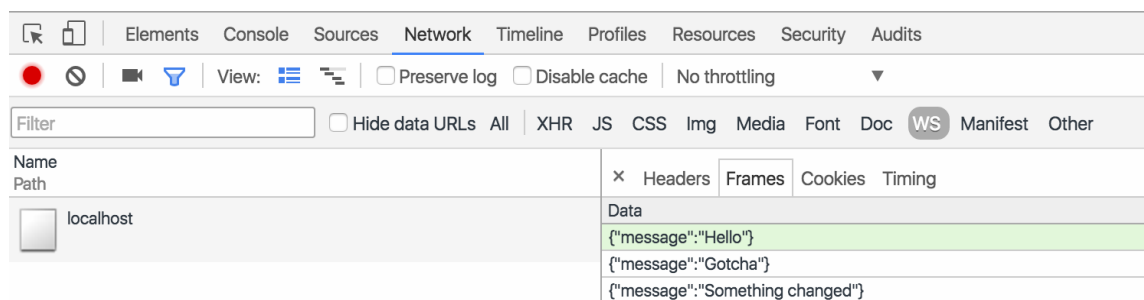


Рисунок 2. – Отладка WebSocket в Chrome

Реализация WebSocket должна присутствовать как на клиентской стороне (по средствам браузерного JavaScript или других языков, транслирующихся в JavaScript, например, Dart), так и на серверной стороне. Наиболее приспособленный для этого серверный язык – NodeJS, в котором реализация WS является встроенной. Примеры кода сообщений на клиентской и серверной части WebSocket приведены в таблице 1.

Таблица 1. – Пример сообщений клиентской и серверной части WebSocket

Client	Server
<pre>let socket = new WebSocket('ws://localhost:8081/'); socket.onopen = function() { let json = JSON.stringify({ message:'Hello ' }); socket.send(json); } socket.onmessage = function(event) { console.log(event.data); } socket.onerror = function(event){ console.log(event); } socket.onclose = function(code, reason) { console.log(code, reason); } window.addEventListener('beforeunload', function() { socket.close();});</pre>	<pre>let WSS = require('ws').Server; let wss = new WSS({ port: 8081 }); wss.on('connection', function(socket) { console.log('Opened'); let json = JSON.stringify({ message: 'Gotcha' }); socket.send(json); socket.on('message', function(message) { console.log('Received: '); }); socket.on('close', function() { console.log('Closed '); }); });</pre>

Заключение

Рассмотренный протокол WebSocket, на основе которого строятся современные realtime веб-приложения, позволяет передавать сообщения в любом количестве без необходимости клиентской части приложения запрашивать эти данные. На основе этой технологии удобно передавать сообщения для чатов, а также другие чувствительные данные по зашифрованному каналу, что не могут конкурирующие технологии. Поэтому для повышения скорости и удобства корпоративной переписки необходимо разработать программный продукт, который будет включать в себя реализацию данной технологии. Данная тема является актуальной, так как комплексного решения этой задачи в настоящее время не существует.

ЛИТЕРАТУРА

1. WebSocket [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/WebSocket>. – Дата доступа: 17.09.2017.
2. WebSockets Web API [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. – Дата доступа: 18.09.2017.

УДК 004.021

РАЗРАБОТКА КОМПЛЕКСНОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПРОЕКТАМИ ИТ-КОМПАНИЙ

А.А. ЛЕВИЦКИЙ

(Представлено: Т.С. РУДЬКОВА)

Предлагается разработка программного обеспечения, созданного для управления проектами ИТ-компаний, которая позволит компаниям управлять своими сотрудниками, а также регулировать задания заказчиков. Анализируются задачи; показан принцип работы и алгоритмы реализации.

При больших темпах роста информационных технологий появляются все больше компаний, которые недобросовестно относятся к своей работе, а также к заказчикам. Заказчики начали тщательнее выбирать исполнителей, вследствие чего молодым компаниям стало сложнее на рынке. Поэтому для более прозрачной работы появилась необходимость в отслеживании работы непосредственно заказчиком. Для более быстрого решения проблем и вопросов, стоящих перед исполнителями и заказчиками, появилась необходимость в чате между каждым из участников проекта.

Так как справляться с наплывами заказов и постоянного роста компаний обычными средствами стало сложнее, то появилась необходимость в разработке централизованной системы для организации слаженной работы команд над разработкой проектов, а также для взаимодействия с заказчиками. Она позволит оценить вклад конкретного разработчика в проект, а также с большим удобством работать исполнителям на удаленной основе.

Анализ задачи. Основным преимуществом сервиса-конкурента «Trello» можно назвать просмотр нескольких одновременно запущенных проектов и их состояние в данный момент. Для работы с группой разработчиков или иных исполнителей, работающих над проектами с конечной датой выполнения или целью, данная система может дать вам представление о ходе выполнения проектов.

Особенностью разработанной системы является то, что каждый пользователь может быстро связаться с другим пользователем посредством чата в реальном времени без необходимости писать непосредственно на корпоративную почту. Например, если заказчик захочет обсудить какие-либо правки, но сделать это нужно очень быстро вследствие занятости, то чат подойдет как нельзя лучше для таких целей. Также модуль позволит разграничить обязанности каждого пользователя.

Выбор языка программирования – один из важнейших аспектов любого проекта. NodeJS – серверная реализация языка программирования JavaScript, основанная на движке V8; предназначена для создания масштабируемых распределенных сетевых приложений, таких как веб-сервер. NodeJS по целям использования сходен с каркасами Twisted на языке Python и EventMachine на Ruby. В отличие от большинства программ JavaScript, этот каркас выполняется не в браузере клиента, а на стороне сервера. Так как NodeJS использует JavaScript синтаксис, то разработчику веб-сайтов легче переключаться между написанием клиентской части, написанной на JavaScript, и серверной частью, написанной на NodeJS. NodeJS является асинхронным языком программирования, построенный на модели событий, что позволяет выполнять код все зависимости от порядка вызовов функций. NodeJS выполняется в одном потоке, но асинхронность достигается путем подписки на события и вызова кластеров [1].

Выбор СУБД пал на документно-ориентированную систему управления базами данных MongoDB с открытым исходным кодом, не требующая описания схемы таблиц. Данная СУБД классифицирована как NoSQL, она использует JSON-подобные документы и схему базы данных. Скорость обеспечивается также тем, что MongoDB написана на языке C++. Запросы могут возвращать конкретные поля документов и пользовательские JavaScript-функции. Поддерживается поиск по регулярным выражениям.

MongoDB может работать с набором реплик, которые состоят из двух и более копий данных. Каждый экземпляр такого набора может в любой момент выступать в роли основной или вспомогательной реплики. Все операции записи и чтения по умолчанию осуществляются с основной репликой, а вспомогательные поддерживают в актуальном состоянии копии данных. В случае, когда основная реплика дает сбой, данный набор проводит выбор, который из реплик должен стать основным. Таким образом, второстепенные реплики могут дополнительно являться источником для операций чтения [2]. Все рассмотренные выше преимущества объясняют выбор MongoDB в качестве СУБД в разрабатываемом приложении.

Принцип работы и алгоритмы реализации. Система разделена на несколько подмодулей (рис. 1), каждый из которых отвечает за отдельный от остальных функционал.

В редактировании профиля можно изменять такие личные данные, как имя, фамилия, электронная почта, номер телефона, а также загружать фотографию, которая будет отображаться в левой панели сайта текущего пользователя, а также в его профиле (рис. 2). Для загрузки фотографии была использована библиотека «jQuery-Fileuploader», которая позволяет просто управлять процессом загрузки на JavaScript.

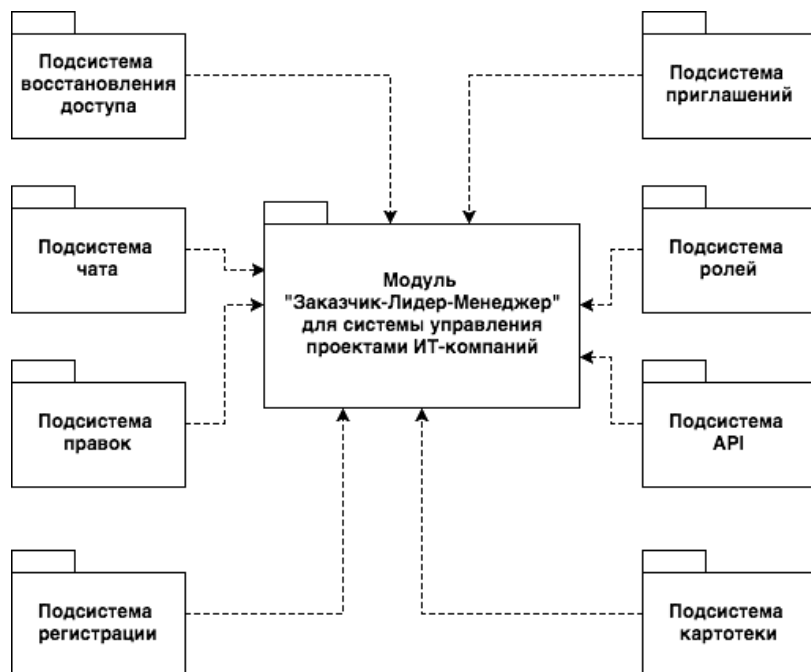


Рисунок 1. – Модули системы управления проектами

Рисунок. 2. – Редактирование профиля

Страница личных сообщений реализована с помощью технологии веб-сокетов. Она позволяет отправлять и принимать любые пакеты данных мгновенно без перезагрузки страницы. Для этого помимо клиентской части также требуется серверная реализация, которая позволит перенаправлять сообщения определенному пользователю в зависимости от полученной информации. Для реализации чата подобного типа была использована библиотека «SocketIO», которая включает в себя реализацию для клиентской

и серверной части веб-приложения. Если у адресата и адресанта открыты чаты друг с другом, то сервер понимает, что эти два пользователя общаются в данный момент. Для большей информативности еще до отправки, при наборе сообщений одним пользователем, другой пользователь видит, что его собеседник набирает сообщение. Это может быть полезно в том случае, если пользователь ждет моментального ответа, но не уверен, пользователь в сети или нет. Для того чтобы увидеть сообщения собеседника, необходимо кликнуть на его имя в левой панели чата. После нажатия будут загружены прошлые сообщения, а также непрочитанные. Все сообщения отсортированы по времени отправления так, что снизу выводятся последние (рис. 3). Слева выводятся сообщения собеседника, справа – сообщения текущего пользователя. По нажатию на фотографию собеседника, пользователь будет переадресован на личную страницу собеседника.

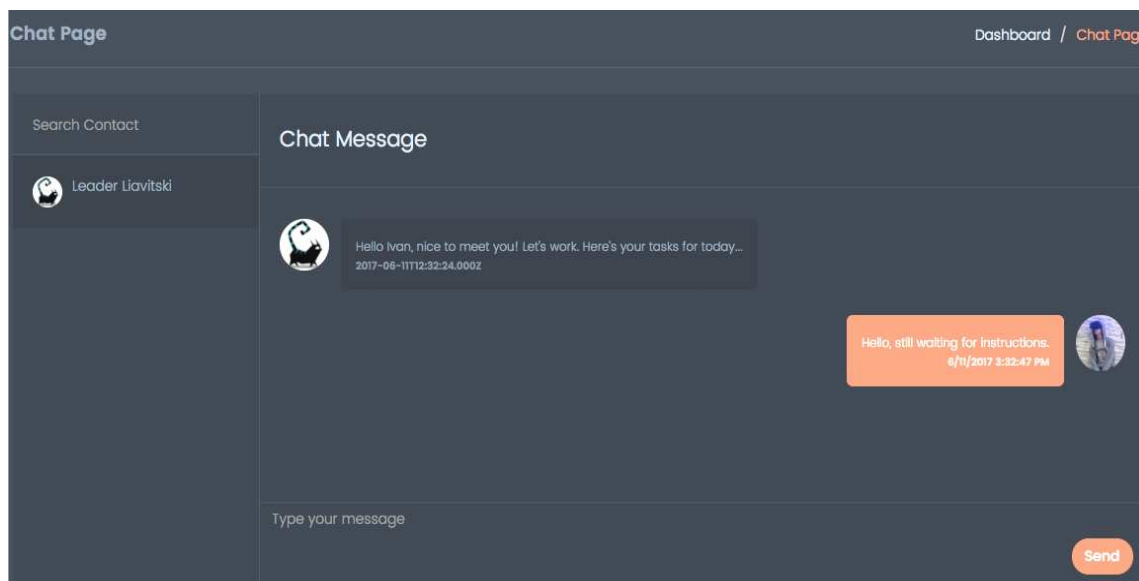


Рисунок 3. – Чат на WebSocket

В разделе «Invites» показаны текущие приглашения, которые еще не были активированы и по которым еще не зарегистрировались приглашенные пользователи. Если пользователь с введенным почтовым ящиком уже существует, то ему не будет отправлено приглашение повторно. При отправлении приглашения генерируется уникальная ссылка, при переходе на которую пользователь сможет зарегистрироваться. За один раз можно добавить неограниченное число сообщений, нажимая кнопку «Add more invites», при необходимости добавить очередное приглашение, при нажатии на которую добавится еще одна строка для ввода новых данных о пользователе. В таблице указать роль приглашаемого пользователя.

Заключение

Разработанная программная система с дружественным пользовательским интерфейсом как для исполнителей, так и для заказчиков позволит компаниям эффективно управлять заданиями заказчиков, эффективно вести переписку, назначать исполнителей, а также точно знать, какую именно задачу выполняет работник.

ЛИТЕРАТУРА

1. Node Package Manager [Электронный ресурс]. – Режим доступа: <https://npmjs.com>. – Дата доступа: 17.09.2017.
2. Базы данных [Электронный ресурс]. – Режим доступа: <http://www.bazydannyyh.ru/>. – Дата доступа: 18.09.2017.

УДК 004.4(004.773.3)

**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ПРИМЕНЯЕМОЕ ДЛЯ СБОРА
КОНФИДЕНЦИАЛЬНОЙ ИНФОРМАЦИИ СРЕДСТВАМИ ЭЛЕКТРОННОЙ ПОЧТЫ****А.В. ЛОБАНОВ***(Представлено: Е.С. ГАТИХО)*

Рассматриваются основные способы и механизмы отслеживания электронных сообщений. Приводятся примеры программного обеспечения, используемого для получения конфиденциальной информации средствами электронной почты.

В настоящее время почти каждый пользователь сети Интернет использует электронную почту для регистрации на различных сервисах общения, обмена корпоративной информацией и для прочих нужд. Сложно представить Интернет без электронных писем. Электронная почта – малозатратный и удобный способ связи в наше время. Однако она может быть применена для доступа к личным данным пользователя [1].

С каждым годом появляется все больше сервисов, направленных на отслеживание действий пользователей и на получение информации о получателях электронных писем.

Такие системы используют в своей работе следующую особенность электронной почты: содержании e-mail сообщения может представлять собой текст, а также может включать HTML. Это означает, что электронное письмо может содержать изображения, видео, файлы стилей и ссылки на различные ресурсы сети Интернет [2].

Принцип работы большинства сервисов отслеживания базируется на внедрении в письмо картинки, которая располагается на внешнем сервере. Когда получатель открывает письмо, идет запрос на почтовый сервис и в ответ приходит сплошной текст, который уже подгружается браузером или клиентским приложением пользователя, в этот момент скачиваются картинки, стили и прочие элементы HTML страницы с внешних серверов, один из которых может быть отслеживающим сервисом.

К основным функциям сервисов отслеживания электронных писем можно отнести:

- получение данных о местонахождении получателя во время открытия письма;
- определение точного времени открытия письма;
- получение данных об устройстве, с которого было прочитано электронное письмо;
- определение пользовательского окружения, такого как: операционная система и ее версия, используемое программное обеспечение для открытия.
- вывод статистики по пользователям, регионами областям [3].

Работа подобных сервисов схематично представлена на рисунке 1.

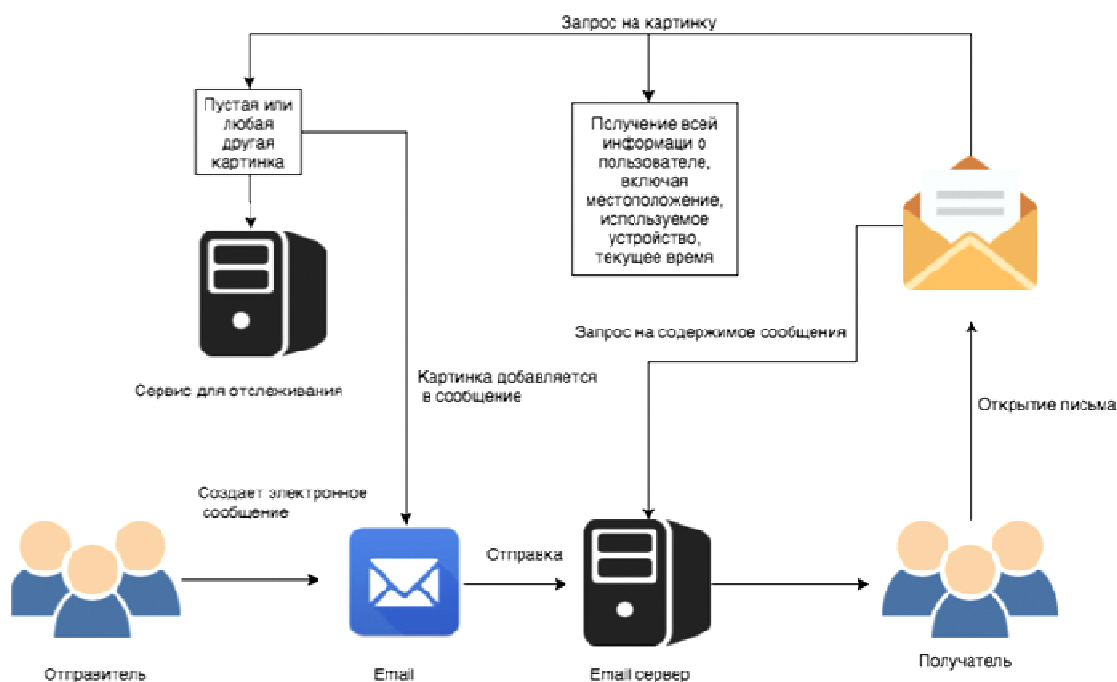


Рисунок 1. – Схема работы систем по отслеживанию сообщений

На текущий момент существует огромное количество сервисов для отслеживания электронных писем, все они предоставляют разный функционал – начиная от отслеживания открытия электронного письма и заканчивая сбором полной информации о пользователе и его действиях, во время открытия или прочтения электронного сообщения.

Список систем самых популярных систем отслеживания:

- Yesware – одна из самых востребованных систем отслеживания электронных писем, разработанная для бизнес-сегмента. Позволяет отслеживать информацию об открытии, кликах, загрузке презентаций и пр. Приложение признано одним из лучших для развития бизнеса по версии Forbes;
- ToutApp – аналог Yeasware;
- Sidekick – система для увеличения продаж с помощью электронных сообщений;
- Mailtrack – популярная система отслеживания сообщений, общее количество скачиваний – около 1 млн., используется профессионалами из Yahoo;
- ContactMonkey – система интеграции систем продажи с Outlook-клиентом, использует систему отслеживания сообщений;
- Boomerang – бесплатная система отслеживания;
- CirrusInsight – популярная система отслеживания сообщений, используется такими компаниями, как LinkedIn, BBC;
- Streak – система отслеживания, используемая Твитером и Uber;
- Outreach – система отслеживания, используемая такими компаниями, как Microsoft, Adobe, Ebay [4].

Заключение

Системы отслеживания контента используются повсеместно, большинство из этих сервисов нужны для сбора маркетинговой информации и не затрагивают конфиденциальную информацию и направлены на улучшение для маркетинговых исследований. Но данный канал связи часто применяется для получения информации о пользователе, в целях отслеживания его местоположения и сбора другой конфиденциальной информации.

ЛИТЕРАТУРА

1. Электронная почта [Электронный ресурс]. – Режим доступа: [http://www.tadviser.ru/index.php/статья:электронная_почта_\(e-mail\)/](http://www.tadviser.ru/index.php/статья:электронная_почта_(e-mail)/). – Дата доступа: 20.09.2017.
2. EmailTracking [Электронныйресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Email_tracking. – Дата доступа: 24.09.2017.
3. Howtrackemail. [Электронный ресурс]. – Режим доступа: <https://www.hubspot.com/sales/email-tracking-guide>. – Дата доступа: 18.09.2017.
4. BestEmailTrackingServices [Электронныйресурс]. – Режим доступа: <https://adespresso.com/blog/5-best-email-tracking-services-know-when-your-email-has-been-opened/>. – Дата доступа: 25.09.2017.

УДК 004.056.53(004.773.3)

МЕТОДЫ ЗАЩИТЫ ОТ СРЕДСТВ СЛЕЖЕНИЯ ЗА ЭЛЕКТРОННОЙ ПОЧТОЙ**А.В. ЛОБАНОВ***(Представлено: Е.С. ГАТИХО)*

Рассматриваются основные способы и механизмы защиты от средств слежения за электронной почтой. Приводятся примеры технологий обеспечения конфиденциальности сокрытия реальной информации о пользователе во время работы с электронной почтой.

На данный момент остро встает вопрос о поиске средств, стратегий, принципов обеспечения Интернет безопасности и предотвращения несанкционированного получения какой-либо информации о конечном пользователе. Информационная безопасность является необходимым условием развития современного общества. Обеспечение информационной безопасности является одной из приоритетных задач многих государств. Сохранность конфиденциальности получаемой и передаваемой информации – это жизненно важный аспект в современном мире. Пользователю необходимо иметь средства защиты личных данных.

Существует несколько подходов по обеспечению защиты пользователя от почтовых сервисов слежения:

- отключение отображения изображений в электронном письме;
- ручное ограничение доступа для определенных адресов, используемых сервисами отслеживания [1].

Ни один из этих вариантов не является универсальным. При отключении изображений большая часть писем потеряет презентабельный вид, а также пользователь может вообще не увидеть необходимой важную информацию.

Во втором случае для каждого конкретного сервиса слежения необходимо собственноручно искать актуальные адреса, которые постоянно обновляются. Также стоит отметить, что количество сервисов постоянно растет. Поэтому этот вариант защиты представляется неудобным для обычного пользователя электронной почты [2].

Основная задача системы для защиты от слежения за электронной почтой состоит в сокрытии, подмены реальной информации о пользователе, чтобы сделать работу отслеживающих систем бессмысленной или невозможной. Самым логичным решением на первый взгляд является подмена изображения, но существует проблема при данном подходе – изменение содержимого сообщения возможно только в сторонних почтовых клиентах.

Система, которая бы могла полностью защищать пользователей во время работы с электронной почтой, должна удовлетворять следующим критериям:

- работа со всеми современными браузерами, так как большая часть пользователей использует только их при работе с электронной почтой;
- мобильные приложения для всех популярных платформ. По статистике, после браузеров, самым популярным методом для работы с электронной почтой являются мобильные устройства;
- расширения для популярных клиентов для работы с электронной почтой.

Если программное обеспечение удовлетворяет всем перечисленным качествам, она может гарантированно защитить пользователя от средств слежения, независимо от используемых платформ.

В случае с работой системы с расширениями для браузера или другими клиентами существует механизм защиты, основанный на перехватах загрузки изображений, и подмены отслеживаемого контента [3]. В момент загрузки всех изображений и другого контента он должен быть просканирован на соответствие определенным маркерам, которые, в свою очередь, указывают на то, что данный контент является отслеживаемым [4]. В этот момент возможно два варианта развития событий:

- изменение содержимого контента, либо полный отказ от его загрузки;
- передача контента другим пользователям в реальном времени и полная загрузка его на клиенте.

Первый способ обеспечивает высокий уровень защиты, но потенциально искажает конечное электронное сообщение. Второй же способ позволяет конечному пользователю скрыться среди множества других пользователей, которые загрузят контент по той же ссылке.

Основные *преимущества* применения разработанной системы:

- возможность прочтение электронной почты без видимых изменений структуры документа;
- обеспечение конфиденциальности данных.

Примерная схема защиты от систем отслеживания электронной почты изображена на рисунке 1.

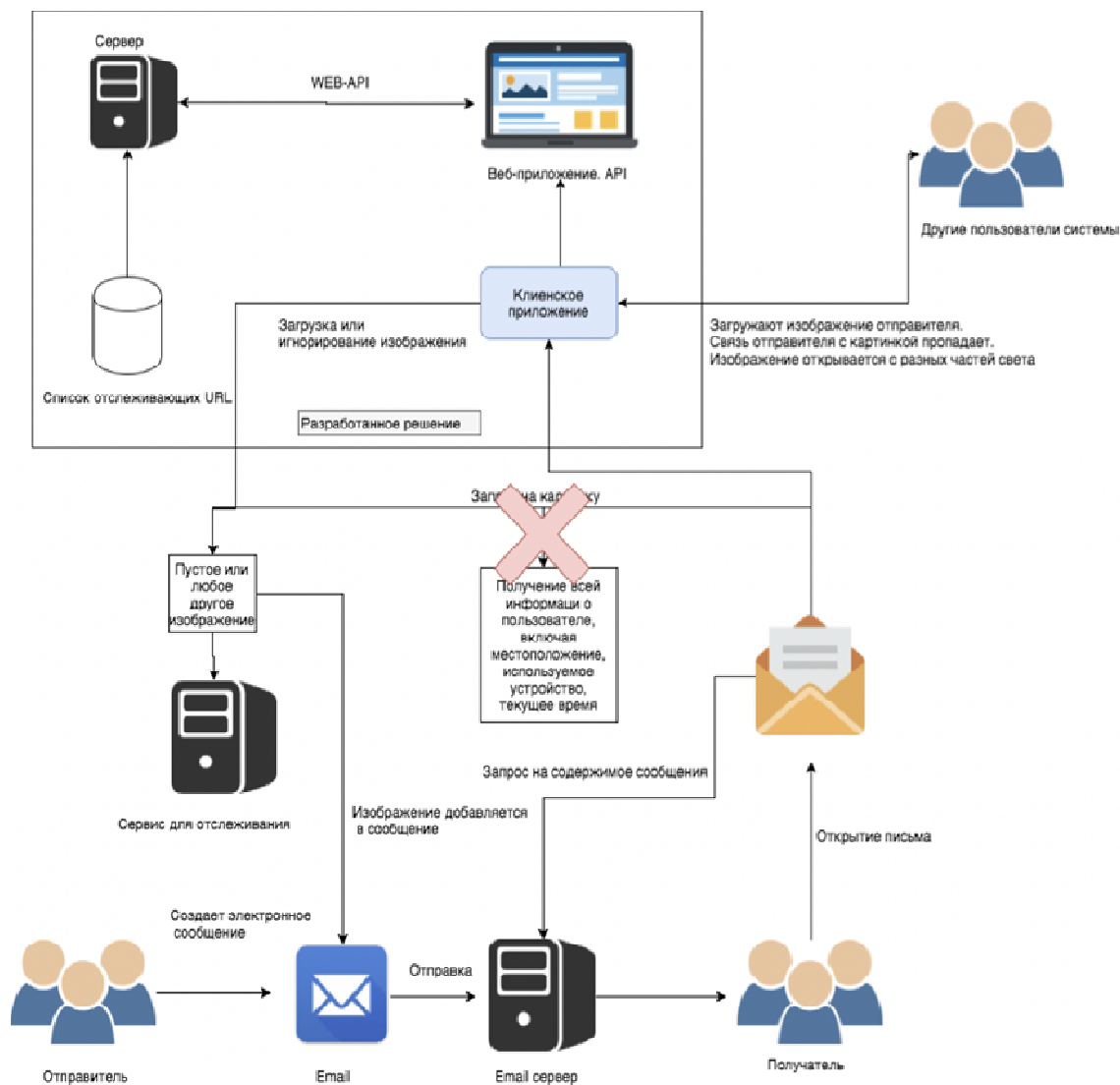


Рисунок 1. – Схема защиты от слежения за электронной почтой

ЛИТЕРАТУРА

1. Howstopemailtracking [Электронныйресурс]. – Режим доступа: <http://www.wikihow.com/Stop-Email-Tracking>. – Дата доступа: 20.09.2017.
2. Blockemailtracking [Электронныйресурс]. – Режим доступа: <https://nordvpn.com/blog/how-to-block-email-tracking/>. – Дата доступа: 24.09.2017.
3. Prevent loading images in browser [Электронныйресурс]. – Режим доступа: <http://www.thewindowsclub.com/disable-images-chrome-firefox-ie>. – Дата доступа: 18.09.2017.
4. Howtrackemailserviceswork [Электронныйресурс]. – Режим доступа: <http://blog.rocketbolt.com/how-does-email-open-tracking-work/>. – Дата доступа: 25.09.2017.

УДК 004.021

**МОБИЛЬНАЯ БАЗА ДАННЫХ REALM КАК АЛЬТЕРНАТИВА SQLITE И COREDATA
ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ IOS****Д.А. МЕДЮХО****(Представлено: Т.С. РУДЬКОВА)**

Анализируются теоретические сведения об особенностях мобильной базы данных Realm. Проведены синтетические тесты быстродействия между Realm, SQLite и CoreData с использованием итерационных и количественных методик.

Одной из главных проблем, с которой сталкиваются многие разработчики, является выбор базы данных. iOS-разработчик обладает небольшим выбором в этом вопросе: SQLite или CoreData. Однако существует еще одно незаурядное решение – мобильная база данных Realm. Она предоставляет разработчикам простой и понятный путь для хранения данных и запросов к ним.

Что такое Realm? Realm – кроссплатформенная мобильная база данных для iOS, доступна как версия для языка Swift, так и для Objective-C [1]. Realm создана для того, чтобы стать лучше SQLite и CoreData. Она не только лучше и быстрее, но и проста в использовании, так как вы можете сделать многое с помощью всего нескольких строк кода. Realm является совершенно бесплатной, и ее использование возможно без каких-либо ограничений. Realm создана для мобильных приложений и это новинка в сегменте мобильных баз данных, так как в последнее десятилетие инноваций в этой области замечено не было. Раньше у вас был только один вариант, которым является SQLite или любая другая обертка, которая использует SQLite.

Realm отличается простой в использовании, так как не является объектно-реляционной, а использует свои собственные наработки для достижения большей производительности и скорости выполнения.

Особенности Realm. Рассмотрим все преимущества и причины, почему использование Realm является лучшим выбором при работе с базой данных в мобильном приложении:

1. *Простая в установке.* Установить Realm легко – с помощью простой команды в Cocoapods, вы перейдете к работе с Realm.

2. *Быстрая.* Realm – невероятно быстрая библиотека для работы с базой данных. Realm быстрее, чем SQLite и CoreData, и сравнительные тесты здесь – лучшее доказательство для этого.

3. *Кроссплатформенная.* Файлы базы данных Realm кроссплатформенные и могут совместно использоваться iOS и Android.

4. *Масштабируемая.* Очень важно рассмотреть масштабируемость при разработке мобильного приложения, если приложение работает с большим количеством пользователей и большим количеством записей. Realm готова к масштабируемости и работе с большими объемами данных в кратчайшие сроки. Вы ускорите ваше приложение с помощью Realm.

5. *Хорошо документированная и есть отличная поддержка.* Команда Realm предоставила читаемую, хорошо организованную документацию о Realm. Если у вас возникли проблемы, вы можете связаться с ними через Twitter, Github или Stackoverflow.

6. *Надежная.* Realm используется большими компаниями в их мобильных приложениях, таких как Amazon, Google и Netflix [2].

7. *RealmBrowser.* Важный инструмент, который стоит внимания – это браузер Realm. Этот браузер помогает читать и редактировать Realm файлы. Эти файлы создаются в приложении со всей информацией о сущностях, атрибутах и записях в таблицах базы данных. Эти файлы могут использоваться на различных платформах, таких как iOS или Android.

8. *Бесплатная.* Все эти удивительные функции абсолютно бесплатны.

Сравнение быстродействия. Испытания проводились на реальной базе данных размером 150 000 записей. Realm оказалась в 2–3 раза быстрее SQLite и CoreData [3].

Синтетические тесты проводились с использованием iPhone SE на операционной системе iOS 11. Тестируемый набор данных – база данных, состоящая из одной таблицы и 150 000 записей.

Результаты прохода по всем записям запроса представлены на рисунке 1.

Результаты получения количества записей показаны на рисунке 2.

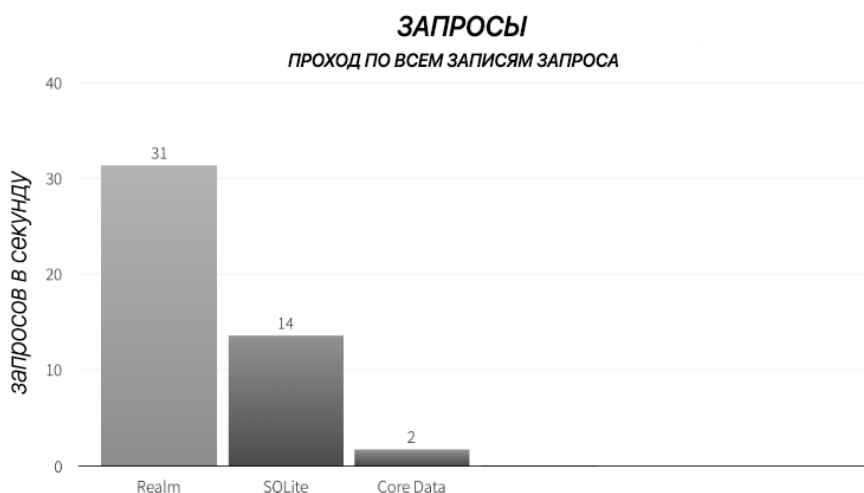


Рисунок 1. – Проход по всем записям запроса

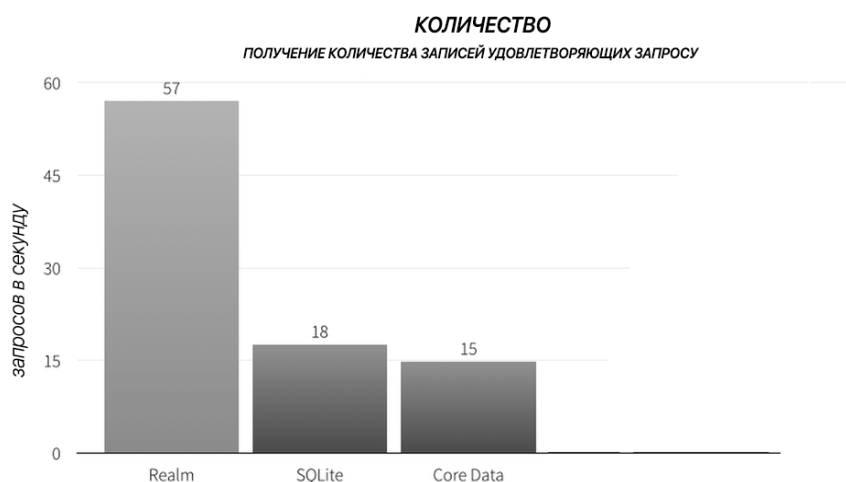


Рисунок 2. – Получение количества записей запроса

Как видно из итерационных и количественных методик тестирования, Realm показывает себя с лучшей стороны и оказывается быстрее своих конкурентов в 2–3 раза. В совокупности с простыми методами использования это дает мощный и одновременно легкий инструмент разработки [3].

Заключение

Realm является легким и простым решением для хранения данных. Realm обладает способностью масштабировать и упрощает работу всего за несколько строк кода, к тому же она быстрее SQLite и CoreData в 2–3 раза. Realm подойдет для большинства приложений и даже игр.

ЛИТЕРАТУРА

1. Realm // Wikipedia [Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/Realm>. – Дата доступа: 17.09.2017.
2. Realm [Электронный ресурс]. – Режим доступа: <https://realm.io/docs/objc/latest/>. – Дата доступа: 17.09.2017.
3. Realm Performance [Электронный ресурс]. – Режим доступа: <https://academy.realm.io/posts/realm-api-optimized-for-performance-and-low-memory-use>. – Дата доступа: 18.09.2017.

УДК 004.021

ШИФРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ДАННЫХ В БАЗЕ ДАННЫХ REALM ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ IOS

Д.А. МЕДЮХО

(Представлено: Т.С. РУДЬКОВА)

Анализируются теоретические сведения о методах защиты пользовательских данных в базе данных Realm. Рассмотрен практический подход использования шифрования и защищенного хранения ключей шифрования.

Каждую секунду пользователи со всего мира вводят личную информацию в установленные приложения, которые неявно обязуются хранить эту информацию в безопасности. Большое количество компаний и разработчиков по-прежнему относятся к безопасности, как аспекту, которым занимаются после реализации основных функций приложения, а не как к основному важнейшему компоненту их программного продукта.

Метод шифрования. В Realm имеется возможность шифрования файлов. Используя 64-байтовый ключ шифрования методом AES-256+SHA-2, Realm становится настолько безопасным, что многие банки начали использовать его в своих клиентских приложениях [1].

AES (Advanced Encryption Standard) – симметричный алгоритм блочного шифрования, принятый правительством США в качестве стандарта в результате конкурса, проведенного между технологическими институтами [2]. Он заменил устаревший DES (Data Encryption Standard), который больше не соответствовал требованиям сетевой безопасности, усложнившимся в XXI веке. Считается, что используемый в AES ключ длиной в 256 бит – достаточно надежная защита против лобовой атаки, то есть с чисто математической точки зрения подобрать один правильный пароль из всех возможных – трудно осуществляемая задача. Несмотря на некоторые недостатки AES, взломать защищенную с помощью этого алгоритма информацию практически нереально. Схема алгоритма показана на рисунке 1.

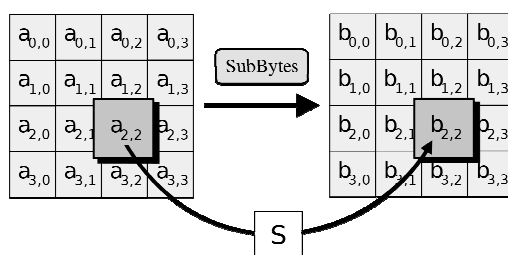


Рисунок 1. – Алгоритм AES Винсента Рэймена и Йона Даймена

Хэш-функции семейства SHA-2 пользуются высокой популярностью в приложениях, связанных с систематизацией, поиском и защитой информации [3]. Криптографические хэш-функции – это математические операции, выполняемые с цифровыми данными; сравнивая хеш (результат выполнения алгоритма) с известным значением хеша, человек может определить целостность данных. Ключевым аспектом криптографических хэш-функций является их устойчивость к взлому: никто не должен иметь возможность найти два разных входных значения, которые приводят к одному и тому же результату хэш-функции. SHA-2 справляется с этим на отлично и был выбран как стандарт хеширования данных. Схема алгоритма представлена на рисунке 2.

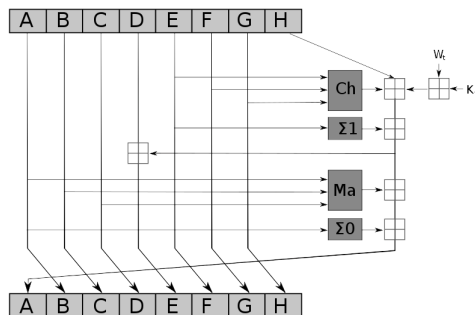


Рисунок 2. – Алгоритм SHA-2

Реализация. Включение шифрования в Realm настолько просто, что требуется только одна дополнительная настройка в настройках конфигурации Realm. Для того чтобы включить шифрование для файлов Realm, необходимо определить свойство *encryptionKey* при создании Configuration базы данных, а затем использовать созданную конфигурацию при создании объекта Realm [2]. Чтобы создать файл с шифрования, необходимо придерживаться следующих шагов:

- 1) генерация 64 байта случайных данных для ключа шифрования;
- 2) создание объекта конфигурации с ключом шифрования;
- 3) открытие файла Realm с созданной конфигурацией;
- 4) обработка исключений при открытии файла с неверным ключом;
- 5) использование объекта для дальнейших запросов.

После создания файла Realm с ключом шифрования, этот ключ будет необходим при каждом новом открытии. При попытке открыть файл с неверным ключом, Realm просто вернет ошибку, указав, что база данных недействительна.

Ограничения. Ключ шифрования для определенного файла Realm всегда только один и его изменение невозможно. Чтобы изменить ключ шифрования, можно экспортировать копию файла Realm, используя *writeCopyToPath(_:encryptionKey:)* и установить новый ключ на копию [2]. Также важно заметить, что шифрование в Realm увеличивает вашу базу данных на небольшой размер из-за необходимых операций шифрования и дешифрования.

Принцип защиты ключа шифрования. Хотя сами данные безопасно защищены Realm, самая важная проблема безопасности – защита ключа шифрования. Ответственность за хранение ключа шифрования несет разработчик. К этому вопросу должны относиться с тем же вниманием, что и к обработке пароля учетной записи пользователя. Самым безопасным вариантом хранения ключа шифрования является его сохранение Keychain. Keychain – надежно защищенный стандарт хранения паролей на устройствах пользователя [5].

Когда зашифрованный файл Realm скопирован на новое устройство, ключ шифрования также должен быть доступен на новом устройстве. Существует много способов, например, сохранить ключ шифрования в Keychain и синхронизировать Keychain через iCloud или через ваш собственный сервис при входе в систему [6].

Заключение

Процесс шифрования считается трудоемким и сложным, но с помощью Realm защита пользовательских данных становится простым шагом. Благодаря тому, что шифрование настолько легко интегрируется, это должно побудить разработчиков правильно защитить данные своих пользователей и свести к минимуму утечки данных в будущем.

ЛИТЕРАТУРА

1. Realm // Wikipedia [Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/Realm>. – Дата доступа: 17.09.2017.
2. Realm [Электронный ресурс]. – Режим доступа: <https://realm.io/docs/objc/latest/>. – Дата доступа: 17.09.2017.
3. Advanced Encryption Standard [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard. – Дата доступа: 17.09.2017.
4. SHA-2 [Электронный ресурс]. – Режим доступа <https://ru.wikipedia.org/wiki/SHA-2>. – Дата обращения 17.09.2017.
5. Adding Capabilities. iCloudKeychain Sharing [Электронный ресурс]. – Режим доступа: https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppDistributionGuide/Adding_Capabilities/Adding_Capabilities.html. – Дата доступа: 18.09.2017.
6. KeychainAccess [Электронный ресурс]. – Режим доступа: <https://support.apple.com/en-us/HT204085>. – Дата доступа: 18.09.2017.

УДК 004.223.2

**ПОДХОДЫ К РАЗРАБОТКЕ ИНТЕРФЕЙСА ДЛЯ ВЕБ-ПРИЛОЖЕНИЯ
ПО РАБОТЕ С МЕДИЦИНСКИМИ АМБУЛАТОРНЫМИ ДАННЫМИ ПАЦИЕНТОВ****Е.А. МЕНИЦКИЙ***(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)*

Представлен практический способ создания интерфейса для работы с медицинскими амбулаторными данными. Проведен анализ технологий, наиболее подходящих для разработки данной системы. Рассмотрена задача, реализация прототипа интерфейса на основе веб-технологий.

На сегодняшний день любое веб-приложение должно быть основано на общих поддерживаемых стандартах разработки, а также обладать понятным для пользователя интерфейсом. Разрабатываемое веб-приложение для работы с медицинскими амбулаторными данными пациентов позволяет производить работу с медицинскими данными, хранящимися в амбулаторных картах, а также удовлетворять требованиям к предоставляемым данным, по видам хранящейся информации и ее представлению.

В данном приложении процесс создания интерфейса приложения служит для прямого взаимодействия приложения с базой данных, сочетающей в себе всю необходимую информацию для вывода непосредственно в приложение. Интерфейс создается путем добавления необходимых компонентов и последующей их настройки. В интерфейсе любого приложения есть компоненты, обеспечивающие взаимодействие программы с пользователем.

Выбор технологий разработки интерфейса

Данное веб-приложение планируется разрабатывать с помощью языка программирования Java и фреймворка веб-разработки Spring Framework. Отсюда следует, что технологии разработки веб-интерфейса должны быть совместимы с данными технологиями, при этом предоставлять все возможности современных интерфейсов.

Разрабатываемое приложение по своей структуре и основной потребности должно давать возможности по работе и представлению большого количества данных, которые бы могли динамически меняться, при действиях пользователя. Анализ инструментов для создания веб-интерфейса позволил выявить несколько технологий, подходящих под данные цели:

JSP (Java Server Pages) – технология, позволяющая веб-разработчикам создавать содержимое, которое имеет как статические, так и динамические компоненты. Страница JSP содержит текст двух типов: статические исходные данные, которые могут быть оформлены в одном из текстовых форматов HTML, SVG, WML, или XML и JSP-элементы, которые конструируют динамическое содержимое. Кроме этого, могут использоваться библиотеки JSP-тегов, а также EL (Expression Language) для внедрения Java-кода в статичное содержимое JSP-страниц [1].

JSF (JavaServer Faces) – это фреймворк для веб-приложений, написанный на Java. Он служит для того, чтобы облегчать разработку пользовательских интерфейсов для Java EE-приложений. В отличие от прочих MVC-фреймворков, которые управляются запросами, подход JSF основывается на использовании компонентов. Состояние компонентов пользовательского интерфейса сохраняется, когда пользователь запрашивает новую страницу и затем восстанавливается, если запрос повторяется [2].

Обе технологии поддерживают возможность работы с динамическими данными, однако предпочтение было отдано технологии Java Server Pages, так как ее поддержка ведется наиболее интенсивно, а также данная технология имеет большую совместимость с Spring Framework благодаря работе через Servlet.

Также было решено добавить возможность адаптивности интерфейса для разрабатываемого приложения. Для этого использовали Bootstrap.

Bootstrap – свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографии, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая Java Script-расширения.

Bootstrap использует современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров [3].

Проектирование интерфейса

При разработке интерфейса, для удобства использования была применена вкладочная структура – информация логически разделена по вкладкам, что упрощает ее поиск, не загромождает главную страни-

цу и не приводит к необходимости создавать дополнительные формы, а также уменьшает время на загрузку других страниц (рис. 1).

Амбулаторная карта Пациенты

№ Карты	Участок	ФИО
2	30	Сергей Петров Сергеевич
3	45	Инна Сидорова Ивановна
5	45	Наталья Шланова Андреевна
6	3	Евгений Терпагин пилл
7	45	Иван Керчин Иванович
11	16	Александр Тетрович Сергеевич
12	16	Альмир Бажанов Сергеевич
13	1	Ольга Бажанова Сергеевна
14	15	Инга Воронова Андреевна
15	16	Серафима Жиглова Александровна
16	17	Семей Жорж Иванович

Номер карты: 11
Участок: 16
ФИО: Александр Тетрович Сергеевич

Создать карту Редактировать Удалить

МИНИСТЕРСТВО ЗДРАВООХРАНЕНИЯ РБ

Форма по ОКУД
Учреждения по ОКПО
Медицинская документация
Форма ... 025/у

МЕДИЦИНСКАЯ КАРТА АМБУЛАТОРНОГО БОЛЬНОГО № 11

Фамилия, имя, отчество: Александр Тетрович Сергеевич
Пол: мужской Дата рождения: 1996-01-01 Телефон: 425678

Адрес больного: область: 16 населенный пункт: Могилев
район: Полоцкий улица (переулок): ул.Садовая
дом №: 16 корпус: 1 кв. №: 20

Место службы, работы: ОАО "МАЗ" отделение, цех: маркетинга

Профессия, должность: продавец

Лист диспансерного наблюдения Лист заключений Профилактические осмотры Лист вакцинации
Рентгелогические обследования Осмотр гинекологом Врачебные заключения Анамнез

Рисунок 1. – Прототип главной страницы для врача

Основная информация о пациентах хранится в таблицах, с которыми имеется возможность производить действия после выбора соответствующих данных. Такая структура является наиболее простой и понятной для пользователей с различным уровнем владения персональным компьютером. Использование стандартных элементов управления позволяет максимально снизить время обучения работе с приложением, а стилизация их с помощью шаблонов Bootstrap позволяет улучшить внешний вид и привлекательность данного приложения.

Также на странице были применены возможности загрузки и просмотра медицинских медиаданных. Осуществлялось это на тех же страницах, где происходит выбор данных пациентов. Данный подход реализован с помощью динамической загрузки данных JSP (рис. 2).

id	Дата	Тип	Доза(р/ч)	Результат	Подтверждение
0	2017-06-08	рентген стоп	10	перелом левой стопы	true
0	2017-06-23	рентген грудной клетки	10	легкие поля чистые	true

id: 0 Дата: 23.06.2017 Тип: рентген стоп Доза (р/ч): 10

Результат: перелом левой стопы

Подтверждение: ☒

Добавить Редактировать Удалить Сохранить в XLS

L R

Рисунок 2. – Прототип страницы с медиа данными

Заключение

Правильным подходом при разработке веб-интерфейса является использование современных технологий, которые позволяют решать свои задачи во всех браузерах. Это, прежде всего, экономит время при разработке интерфейса, а также ресурсы на обработку данных, что является актуальным при большом количестве информации, которая подлежит постоянному изменению.

Рассмотренные технологии JSP и Bootstrap при взаимодействии позволяют создать адаптивный интерфейс любого уровня сложности, при этом сохраняя его функциональные возможности и привлекательный внешний вид. По итогам данной работы описана возможность их применения к построению интерфейса и динамической обработке запросов без перезагрузки страницы.

ЛИТЕРАТУРА

1. Java Server Pages // Википедия : свободная энцикл. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/JSP>. – Дата доступа: 25.09.2017.
2. Java Server_Faces // Википедии : свободная энцикл [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/JavaServer_Faces . – Дата доступа: 25.09.2017.
3. Bootstrap // Википедия : свободная энцикл. [Электронный ресурс]. – Режим доступа: [https://ru.wikipedia.org/wiki/Bootstrap_\(фреймворк\)](https://ru.wikipedia.org/wiki/Bootstrap_(фреймворк)). – Дата доступа: 25.09.2017.

УДК 004.223.2

**ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ
С МЕДИЦИНСКИМИ АМБУЛАТОРНЫМИ ДАННЫМИ ПАЦИЕНТА****Е.А. МЕНИЦКИЙ***(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)*

Рассматривается проектирование системы для работы с медицинскими амбулаторными данными, а также вопросы защиты этих данных. Проведен анализ технологий, наиболее подходящих для разработки данной системы. Проведены исследования по актуальности разработки данной системы.

На данный момент в Республике Беларусь, множество учреждений здравоохранения используют бумажный документооборот для хранения амбулаторных данных пациентов. Данный подход с момента развития информационных технологий значительно уступил в эффективности электронным медицинским системам хранения и обработки медицинских данных пациентов. Электронные системы имеют ряд преимуществ [1]:

- повышение качества управления медицинской, административной и финансовой деятельностью организации здравоохранения за счет комплексного внедрения электронной медицинской карты и электронного талона, предоставления дополнительных электронных услуг автоматического самообслуживания для пациентов;
- сокращение времени оформления медицинской документации, посредством получения готовых данных из баз данных и компьютерного ввода;
- сокращение сроков предоставления отчетов и реестров услуг на основе автоматически генерируемых шаблонов;
- оптимизация штата медицинского персонала учреждений здравоохранения;
- обеспечение оперативного контроля за показателями работы поликлиники, экспертизы качества оказания медицинской помощи, посредством логирования данных и составления отчетной выборки;
- повышение достоверности информации о показателях лечебно-диагностической работы и финансовой деятельности посредством методов криптографической защиты информации;
- повышение пропускной способности учреждений здравоохранения за счет сокращения времени работы с амбулаторной картой и другими документами, и как следствие – сокращение очередей;
- предоставление возможностей телемедицины, для коммуникации по врачам и пациентам, а также консультации врача с врачом.

Данные критерии, несомненно, доказывают эффективность электронных систем, а также актуальность их разработки и внедрения в систему здравоохранения.

Средства решения задачи

Основу разрабатываемой системы составляет возможность хранить, накапливать и обрабатывать данные. Медицинские данные должны храниться в строго систематизированном виде, основываясь на нормах ведения медицинской документации, и иметь явную взаимосвязь.

Основываясь на этих данных, необходимо будет использовать реляционную базу данных для хранения сущностей и язык запросов SQL для построения базы данных. В данном случае использование именно реляционной базы данных позволит формально и однозначно определить ограничения целостности БД в терминах ее концептуальной схемы.

Так как медицинские данные строго регламентированы, при разработке системы целесообразным будет использовать официальные документы по составлению амбулаторных медицинских карт [2] и представлять отдельный лист вида обследования или сбора данных, как отдельную сущность, связанную с пациентом.

Очевидно, что при реализации приложения важно сформировать некоторую архитектуру. Для взаимодействия с базой данных подойдет паттерн MVC, представляющий собой взаимодействие 3-х элементов: модель, представление и контроллер [3]. Модель является некоторыми данными, которые описывают предметную область, и над ними могут быть выполнены некоторые операции. Также эта информация может быть предоставлена пользователю на графическом интерфейсе.

Представление и является этим самым графическим интерфейсом. Через данный интерфейс пользователи системы будут передавать данные на выполнение задачи контроллером. Контроллер, в свою очередь, реализует некоторые алгоритмы для получения данных с представления, обработки их и отправки обратно.

Также одним из преимуществ подобных систем является возможность передачи информации в виде потокового видео-сообщения. Реализация данного функционала позволит проводить видеокон-

сультации и осмотры, что, несомненно, уменьшит время на обслуживание пациента, тем самым позволит разгрузить учреждения здравоохранения. Для реализации данной системы, нам подойдет технология WebRTC.

Технология WebRTC – это технология, позволяющая интернет-приложениям производить захват видео и аудио- и медиапоток опционально, не прибегая к посредническим сервисам и протоколам. Набор стандартов, которые включает в себя технология WebRTC, позволяет обмениваться данными и проводить телеконференции в режиме узел-узел [4].

Общая архитектурная схема веб-приложения для работы с медицинскими амбулаторными данными пациента представлена на рисунке 1.

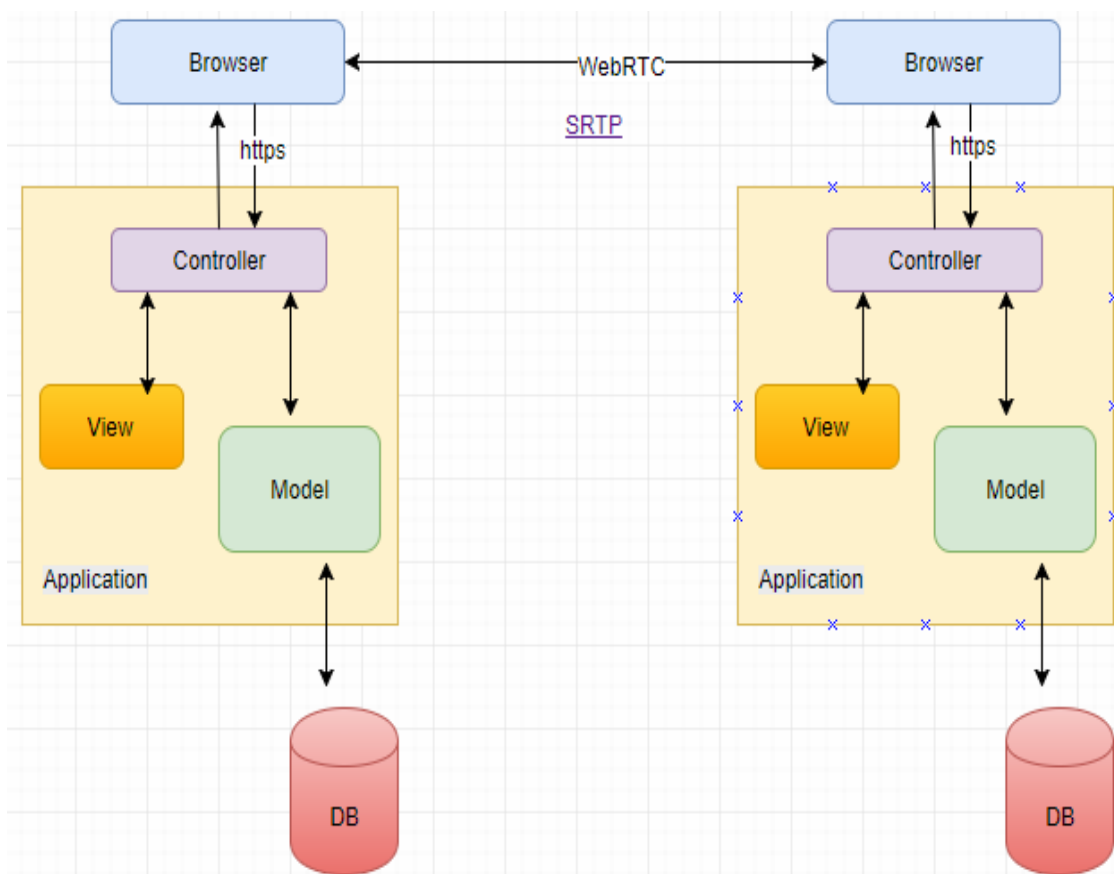


Рисунок 1. – Общая архитектурная схема веб-приложения для работы с медицинскими амбулаторными данными пациента

Принцип защиты информации

Основываясь на том, что данная система предназначена для хранения конфиденциальных данных пациентов, необходимо разработать систему аутентификации, и криптографической защиты данных.

Для предоставления аутентификации, необходимо создать сущность пользователя в базе данных с различными видами ролей, такими как пациент, врач, администратор и т.п., при этом разделить уровни доступа к алгоритмам работы с данными, согласно реальному уровню доступа к медицинской документации. Также разделения прав на использование алгоритмов должно быть реализовано как на уровне базы данных, через редактор ролей, так и через контроллер, с помощью проверок на этапе кодирования.

Для защищенной передачи данных по сети интернет предназначены следующие виды технологии:

- *протокол HTTPS* – стандарт передачи данных между различными машинами, который определяет, что должно выступать в качестве сигнала начала передачи, как обозначаются данные и т.д. При этом данные шифруются по протоколу SSL, что делает проблематичным не только перехват, но и получение конфиденциальной информации [5];

- *протокол SRTP* – это профиль расширения RTP (Real-Time Transport Protocol, транспортный протокол в реальном времени), который добавляет дополнительные функции безопасности, такие как аутентификация сообщений, конфиденциальность и защита от прослушивания. Данный протокол будет использован при передаче видеосообщений [6];

- *шифр AES* – симметричный алгоритм блочного шифрования. Данный вид шифрования будет использовать один и тот же симметричный ключ. Для разрабатываемой системы это является оптимальным вариантом, так как подобные системы также защищены юридическими соглашениями о неразглашении третьим лицам. Используемый метод шифрования будет использован для шифрации конфиденциальных данных на уровне базы данных.

Заключение

В ходе данного исследования спроектирована программная система, наиболее подходящая по архитектурным особенностям под основу для разработки веб-приложения для работы с медицинскими амбулаторными данными пациента. При этом разработанная архитектура является универсальной для любого современного языка программирования, а также оставляет возможности для реализации необходимого функционала, обеспечивая при этом высокий уровень защиты данных.

ЛИТЕРАТУРА

1. Информатизация амбулаторно-поликлинического звена [Электронный ресурс]. – Режим доступа: http://minzdrav.gov.by/ru/static/informatization/inform_apzveno. – Дата доступа: 23.09.2017.
2. Оформление медицинской карты амбулаторного больного [Электронный ресурс]. – Режим доступа: https://www.bsmu.by/downloads/kafedri/k_poli_ter/stud/v1.pdf. – Дата доступа: 23.09.2017.
3. Model-View-Controller [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Model-View-Controller>. – Дата доступа: 23.09.2017.
4. WebRTC API [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/API/WebRTC_API. – Дата доступа: 23.09.2017.
5. Протокол HTTPS – что такое? [Электронный ресурс]. – Режим доступа: <http://fb.ru/article/221368/protokol-https---chto-takoe>. – Дата доступа: 23.09.2017.
6. SRTP – что такое безопасный протокол передачи данных в реальном времени? [Электронный ресурс]. – Режим доступа: <https://www.3cx.ru/web rtc/srtp/>. – Дата доступа: 23.09.2017.

УДК 004.853

БИБЛИОТЕКА ГЛУБИННОГО ОБУЧЕНИЯ DEEPLARNING4J

А.А. СОЛОВЬЁВ

(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)

Анализируется библиотека с открытым исходным кодом, распределенная библиотека глубинного обучения для виртуальной машины Java (JVM) Deeplearning4J (DL4J) на примере архитектуры сверточной нейронной сети, представленной Алексом Крижевским (AlexKrizhevsky) на конкурсе ImageNet в 2012 году.

Стремительное развитие технологий машинного обучения делает спрос на различные инструменты, позволяющие обучать архитектуры нейронных сетей любой сложности, а также тонко настраивать различные параметры при обучении. Одним из современных инструментов машинного обучения является библиотека DL4J.

DL4J – это первая коммерческая библиотека глубинного обучения с открытым исходным кодом, написанная для Java и Scala [1]. Данная библиотека позволяет строить архитектуры нейронных сетей различной сложности и глубины, тонко настраивать всевозможные параметры обучения. Библиотека предназначена для использования в бизнес-средах на распределенных графических и центральных процессорах, включает в себя реализацию ограниченной машины Больцмана, глубокой сети убеждений, глубокого автокодера, многоуровневого автокодирования и рекурсивной сети нейронных тензоров, word2vec, doc2vec и GloVe. Все эти алгоритмы включают распределенные параллельные версии, которые интегрируются с ApacheHadoop и Spark [2].

Для анализа данной библиотеки рассмотрим архитектуру сверточной нейронной сети AlexNet, которая заняла первое место в конкурсе ImageNet в 2012 году [3].

Программная реализация данной архитектуры сверточной нейронной сети при помощи библиотеки DL4J представлена следующим образом:

```
01: MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
02: .seed(seed)
03: .weightInit(WeightInit.DISTRIBUTION)
04: .dist(new NormalDistribution(dist))
05: .activation(Activation.RELU)
06: .updater(Updater.NESTEROVS).momentum(momentum)
07: .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
08: .iterations(iterations)
09: .gradientNormalization(GradientNormalization.RenormalizeL2PerLayer)
10: .learningRate(learningRate)
11: .regularization(true).l2(l2)
12: .list()
13: .layer(0, new ConvolutionLayer.Builder(kernelSize, stride, padding)
    .nIn(in).nOut(out).build())
14: .layer(1, new LocalResponseNormalization.Builder().build())
15: .layer(2, new SubsamplingLayer.Builder(kernelSize, stride).build())
16: .layer(3, new ConvolutionLayer.Builder(kernelSize, stride, padding)
    .nIn(in).nOut(out).build())
17: .layer(4, new LocalResponseNormalization.Builder().build())
18: .layer(5, new SubsamplingLayer.Builder(kernelSize, stride).build())
19: .layer(6, new ConvolutionLayer.Builder(kernelSize, stride, padding)
    .nIn(in).nOut(out).build())
20: .layer(7, new ConvolutionLayer.Builder(kernelSize, stride, padding)
    .nIn(in).nOut(out).build())
21: .layer(8, new ConvolutionLayer.Builder(kernelSize, stride, padding)
    .nIn(in).nOut(out).build())
22: .layer(9, new SubsamplingLayer.Builder(kernelSize, stride).build())
23: .layer(10, new DenseLayer.Builder().nOut(out).dropOut(dropOut).build())
```

```
24: .layer(11, new DenseLayer.Builder().nOut(out).dropOut(dropOut).build())
25: .layer(12, new OutputLayer.Builder(LossFunctions.NEGATIVELOGLIKELIHOOD)
    .nOut(out).activation(Activation.SOFTMAX).build())
26: .backprop(true)
27: .setInputType(InputType.convolutional(height, width, channels))
28: .build().
```

Рассмотрим данный программный код подробнее. В первой строке программного кода происходит объявление и инициализация многослойной нейронной сети. Создается объект класса `MultiLayerConfiguration`, которому присваивается значение нового объекта класса `NeuralNetConfiguration`. Данный класс реализует методы конфигурирования нейронных сетей, которые охватывают параметры каждого слоя.

Далее устанавливаются необходимые нам параметры нейронной сети. Метод `seed` принимает в себя вещественное число, которое используется при инициализации весов нейронной сети. Инициализация весов происходит по определенному закону, который устанавливается в методе `weightInit`. Тип закона устанавливается в методе `dist`. В нашем случае инициализация весов происходит по нормально-распределенному закону.

Также необходимо указать функцию активации для скрытых слоев нейронов. Для этого используется метод `activation`, который в качестве параметра принимает вид функции. В настоящее время огромной популярностью пользуется выпрямленная линейная функция активации `RELU`. Использование данной функции приводит к прореживанию весов, что ограждает нейронную сеть от переобучения.

После в методе `updater` указывается механизм обновления весов. В большинстве случаев хорошим выбором является использование алгоритма оптимизации стохастического градиентного спуска `NESTEROVS` [1]. Метод `momentum` является дополнительным методом, который отвечает за то, насколько быстро алгоритм оптимизации сходится в оптимальной точке.

Метод `iterations` принимает целое число, которое указывает на количество итераций для одной эпохи обучения сети.

Градиентная нормализация (`gradientNormalization`) применяется во избежание проблем малых и больших градиентов.

В методе `learningRate` указывается скорость обучения нейронной сети. Данный параметр является одним из важнейших при обучении сети. Высокая скорость обучения сети означает, что в обновлениях веса будут делаться крупные шаги, поэтому образцу может потребоваться меньше времени, чтобы набрать оптимальный набор весов. Но слишком высокая скорость обучения может привести к крупным и недостаточно точным скачкам, которые помешают достижению оптимальных показателей.

В следующем методе мы указываем нашей сети, что будем использовать регуляризацию.

Регуляризация помогает избежать переобучения нейронной сети. В нашем случае мы используем `L2`-регуляризацию. Регуляризация `L2` накапливает большие весовые коэффициенты сети и предотвращает слишком большой вес. Общие значения для `L2`-регуляризации составляют от $1e-3$ до $1e-6$.

Функция `list` позволяет создать многослойную нейронную сеть. В архитектуре рассматриваемой сети находится: 13 слоев, из которых 5 сверточных слоев; 3 слоя субдискретизации; 2 нормализации; 2 полносвязных слоя и выходной слой.

Метод `layer` принимает два параметра – номер слоя и объект класса `Layer`. Рассматриваемые далее классы слоев наследуются от класса `Layer`.

Класс `ConvolutionLayer` реализует метод `builder`, в котором устанавливаются такие параметры, как размер ядра свертки, шаг свертки и отступ. Также класс реализует методы `nIn` и `nOut`, в которых устанавливаются количество входных и выходных нейронов; метод `build` – конфигурирует заданные параметры.

Класс `SubsamplingLayer` также реализует метод `builder`, в котором указываются размер ядра и шаг субдискретизации. Данный класс поддерживает следующие типы субдискретизации: `MAX`, `AVG`, `NON`. Слои данного типа предназначены для уменьшения размерности карт признаков, полученных после свертки. Уменьшение размерности карт признаков приводит к сокращению параметров, используемых при обучении, без существенной потери важной информации.

Класс `LocalResponseNormalization` реализует алгоритм нормализации данных.

Класс `DenseLayer` реализует функционал полносвязного слоя прямого распространения. Также в данном классе реализован метод `dropOut`, который принимает вещественное значение, указывающее на процент нейронов, которые будут отброшены во время обучения [4]. Данный параметр позволяет сети не переобучаться.

Класс `OutputLayer` реализует функционал вывода данных с различными объективными совпадениями для разных целей. Данный класс производит классификацию, а также регрессию. Метод `Builder`

данного класса принимает в качестве аргумента некоторую функцию. В нашем случае это функция потерь отрицательного логарифмического правдоподобия.

Метод `backprop` реализует механизм обратного распространения ошибки, в качестве параметра принимает значение типа `bool`, которое указывает, использовать ли данный метод при обучении сети.

В методе `setInputType` устанавливается тип нейронной сети (в нашем случае сверточная), а также размер и количество каналов входных изображений.

Метод `build` собирает описанную выше нейронную сеть.

Заключение

В данной работе были рассмотрены лишь некоторые возможности библиотеки DL4J и проанализированы основные моменты при построении сверточной нейронной сети. Достоинствами данной библиотеки являются хорошая документация классов, методов и алгоритмов, огромное количество всевозможных методов обучения нейронных сетей, а также то, что данная библиотека написана для виртуальной машины Java. В библиотеке содержится большое количество методов конфигурирования нейронных сетей, однако основополагающими методами являются методы инициализации весов (`weightInit`), установки функции активации (`activation`) и скорости обучения сети (`learningRate`), а также конфигурирования слоев (`layer`) и способа их обучения. Также стоит отметить, что DL4J – это библиотека с открытым исходным кодом, что позволяет свободно использовать и распространять исходные коды с использованием данной библиотеки.

ЛИТЕРАТУРА

1. Библиотека глубинного обучения DeepLearning4J [Электронный ресурс]. – Режим доступа: <https://deeplearning4j.org/>. – Дата доступа: 23.09.2017.
2. Библиотека с открытым исходным кодом DeepLearning4J [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Deeplearning4j/>. – Дата доступа: 24.09.2017.
3. Krizhevsky, A. Imagenet classification with deep convolutional neural networks / Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton // NIPS. – 2012. – 1106–1114 p.
4. Dropout: A Simple Way to Prevent Neural Networks from Overfitting / Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever // Journal of Machine Learning Research. – 2014. – (15). – P. 1929–1958.

УДК 004.021

**ОБНАРУЖЕНИЕ ДЫМА ОТ ПОЖАРА С ПРИМЕНЕНИЕМ ОБРАБОТКИ
НА ОСНОВЕ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ****А.А. СОЛОВЬЁВ***(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)*

Описывается алгоритм детектирования дыма от пожара, реализованный на языке программирования Java. Приведены примеры работы разработанного приложения и сделаны выводы о целесообразности использования сверточных нейронных сетей для решения поставленной задачи.

В связи с интенсивным развитием вычислительной техники, системы машинного зрения стали доступны широкому классу пользователей и позволяют эффективно решать многие задачи в практической деятельности человека [1; 2]. Одной из актуальных прикладных задач, решаемых в последнее время с помощью систем машинного зрения, является обнаружение лесных пожаров. Для решения данной задачи был разработан алгоритм обработки динамических изображений с применением глубокого обучения, в частности сверточных нейронных сетей (СНС). Разработанный алгоритм включает в себя следующие три стадии выполнения программы: цветовая сегментация, обнаружение движущихся областей и классификация выделенных предполагаемых дымчатых областей.

Локализация объектов в сцене наблюдения происходит в два основных этапа. Сначала в видеопоследовательности осуществляется поиск движущихся объектов, после чего происходит их классификация по цветовым характеристикам.

В качестве метода обнаружения движения был выбран метод вычитания фона, основанный на смеси Гауссовых распределений. Данный метод позволяет обнаруживать медленно движущиеся объекты в сцене.

Классификацию движущихся объектов было решено проводить в цветовом пространстве HSV, так как классификация в данном цветовом пространстве дает меньше ложных обнаружений дыма, чем в цветовом пространстве YUV. Классификатором локализованных областей является обученная сверточная нейронная сеть. В качестве архитектуры нейронной сети выбрана модель LeNet [3], которая является простой, не требующей больших вычислительных ресурсов и позволяющей производить обучение.

Общая схема разработанного алгоритма представлена на рисунке 1.

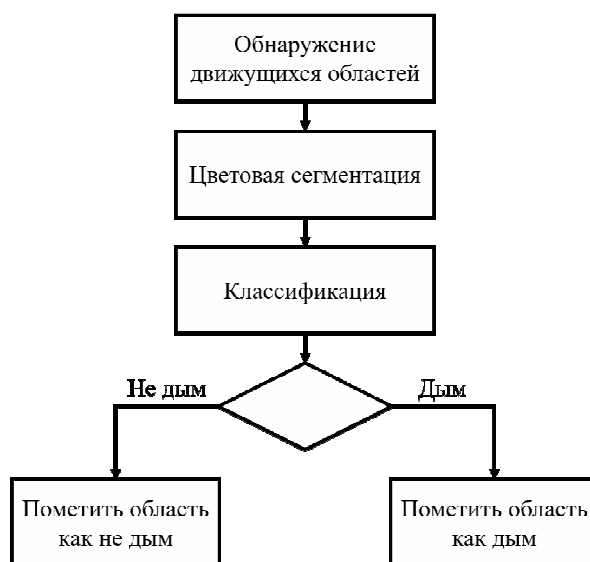


Рисунок 1. – Общая схема разработанного алгоритма

Для проведения экспериментов и оценки результатов исследований разработанный алгоритм был реализован на языке Java с использованием интегрированной среды разработки программного обеспечения IntelliJIDEA, библиотеки компьютерного зрения OpenCV 3.0 и библиотеки глубокого обучения DL4J. При проектировании графического интерфейса было принято решение использовать платформу для создания приложений JavaFX.

Разработка, тестирование и анализ результатов работы алгоритма проводились с помощью ноутбука Samsung 300E5Z со следующими основными характеристиками: центральный процессор – IntelCore i3-2350M с тактовой частотой 2,3 ГГц; объем ОЗУ – 4,0 Гб; графический ускоритель – IntelHDGraphics 3000.

Для захвата видеопотока (с камеры видеонаблюдения либо из постоянной памяти ЭВМ) и извлечения кадров из видеопоследовательности использовалась библиотека OpenCV. OpenCV (OpenSource-ComputerVisionLibrary) – библиотека алгоритмов компьютерного зрения, обработки изображений с открытым исходным кодом [4]. OpenCV создана для обеспечения общего стандартного интерфейса машинного зрения для приложений в области обработки динамических изображений и увеличения использования систем машинного зрения в коммерческих продуктах.

Построение, обучение и использование СНС проводилось при помощи библиотеки DL4J (DeepLearningforJava) – библиотека глубинного обучения, написанная для Java и виртуальной машины Java (JVM), и вычислительная среда с широкой поддержкой алгоритмов глубокого обучения [5].

Для демонстрации возможностей алгоритма было решено реализовать приложение, предназначенное для раннего детектирования лесных пожаров. Интерфейс реализованного приложения представлен на рисунке 2. Интерфейс приложения состоит из главного окна, окна вывода текстовой информации и окна вывода видеoinформации.

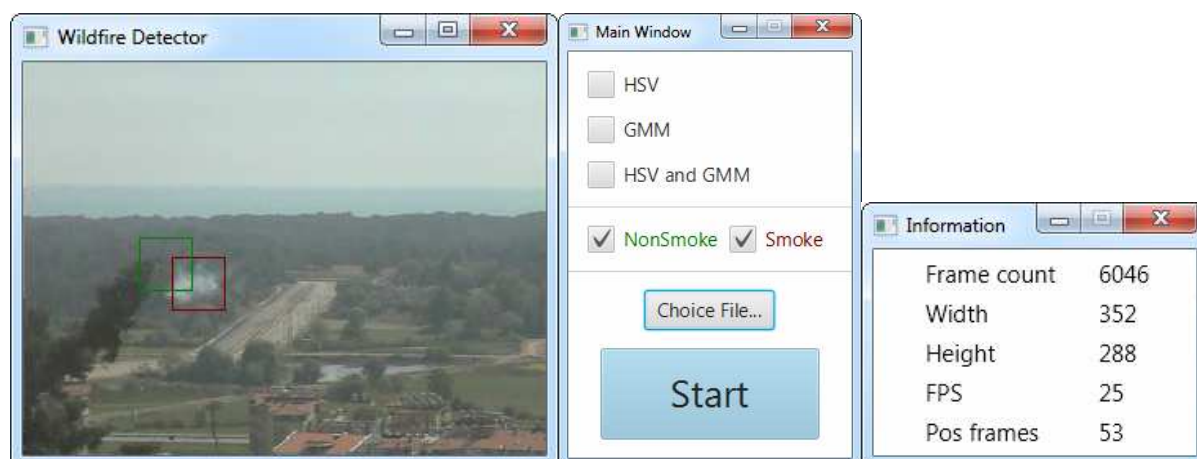


Рисунок 2. – Интерфейс программного продукта

Для анализа разработанного алгоритма использовались видеопоследовательности, полученные в условиях реального освещения при съемке вне помещения с различным углом наклона, в том числе и с нестабильным фоном. Работа разработанного приложения представлена на рисунке 3.



Рисунок 3. – Кадры используемых видеопоследовательностей

Заключение

Результаты тестирования разработанного алгоритма показали, что сверточные нейронные сети могут служить в качестве классификатора пожара. Ввиду того, что в качестве классификатора была выбрана простая модель сети, а при обучении использовалась относительно небольшая выборка примеров, тестирование показало плохие результаты. Коэффициент ложноположительной классификации разработанного алгоритма оказался равен 7%, что недопустимо. Однако можно предположить, что при применении сложных архитектур нейронных сетей и большой обучающей выборки результаты детектирования пожара могут быть улучшены многократно.

ЛИТЕРАТУРА

1. Методы автоматического обнаружения и сопровождения объектов. Обработка изображений и управление / Б.А. Алпатов [и др.] – М. : Радиотехника, 2008. – 176 с.
2. Ngan, K. Video Segmentation and Its Applications / K. Ngan, H. Li // Springer, 2011. – 165 p.
3. LeCun Yann. Gradient-based learning applied to document recognition / Yoshua Bengio, Yann LeCun, Leon Bottou, Patrick Haffner // IEEE, 1998.
4. Библиотека компьютерного зрения OpenCV [Электронный ресурс]. – Режим доступа: <http://opencv.org/>. – Дата доступа: 27.09.2017.
5. Библиотека глубинного обучения DL4J [Электронный ресурс]. – Режим доступа: <https://deeplearning4j.org/>. – Дата доступа: 27.09.2017.

УДК 004.8

СПОСОБЫ МАШИННОГО ОБУЧЕНИЯ

М.А. ШЕВЦОВ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматриваются способы машинного обучения, отдельные из них поддерживают несколько видов обучения, некоторые могут быть реализованы только с определенным алгоритмом обучения. В данной работе рассмотрены основные из рассмотренных способов.

1. Обучение с учителем (англ. **supervised learning**) – один из способов машинного обучения, в ходе которого испытуемая система принудительно обучается с помощью примеров «стимул-реакция».

Между входами и эталонными выходами (стимул-реакция) может существовать некоторая зависимость, но она неизвестна. Известна только конечная совокупность пар «стимул – реакция», называемая обучающей выборкой. На основе этих данных требуется восстановить зависимость (построить модель отношений стимул – реакция, пригодных для прогнозирования), то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ.

Под учителем понимается либо сама обучающая выборка, либо тот, кто указал на заданных объектах правильные ответы.

Постановка данного эксперимента предполагает наличие экспериментальной системы, метода обучения и метода испытания системы или измерения характеристик.

Основные алгоритмы:

- линейная и логистическая регрессия;
- машина опорных векторов;
- наивный Байес;
- нейронная сеть;
- повышение градиента;
- классификационные деревья и случайный лес.

2. Обучение без учителя (англ. **unsupervised learning**) – способ машинного обучения, при котором испытуемая система спонтанно обучается выполнять поставленную задачу без вмешательства со стороны экспериментатора. Метод пригоден для задач, в которых известны описания множества объектов (обучающей выборки) и требуется обнаружить внутренние взаимосвязи, зависимости, закономерности, существующие между объектами.

Обучение без учителя часто противопоставляется обучению с учителем, когда для каждого обучающего объекта принудительно задается «правильный ответ», и требуется найти зависимость между стимулами и реакциями системы.

Экспериментальная схема обучения без учителя часто используется в теории распознавания образов, машинном обучении. При этом в зависимости от подхода формализуется в ту или иную математическую концепцию.

Основные алгоритмы:

- алгоритм кластеризации, такой как K-mean, иерархическая кластеризация или модели смеси;
- алгоритмы сокращения размерности, такие как PCA, ICA или autoencoder;
- обнаружения аномалий, чтобы найти выбросы в данных, т.е. наблюдения, которые не соответствуют шаблонам набора данных.

3. Обучение с подкреплением (англ. **reinforcement learning**) – один из способов машинного обучения, в ходе которого испытуемая система (агент) обучается, взаимодействуя с некоторой средой. Откликом среды (а не специальной системы управления подкреплением, как это происходит в обучении с учителем) на принятые решения являются сигналы подкрепления, поэтому такое обучение является частным случаем обучения с учителем, но учителем является среда или ее модель. Также нужно иметь в виду, что некоторые правила подкрепления базируются на неявных учителях.

Учитывая его состояние и состояние окружающей среды, агент выберет действие, которое максимизирует его вознаграждение или будет исследовать новую возможность. Эти действия изменяют состояние среды и агента. Они также будут интерпретироваться, чтобы дать вознаграждение агенту. Выполняя этот цикл много раз, агенты улучшат свое поведение.

Агент воздействует на среду, а среда воздействует на агента. О такой системе говорят, что она имеет обратную связь. Такую систему нужно рассматривать как единое целое, поэтому линия раздела между средой и агентом достаточно условна.

Основные алгоритмы:

- Q-Learning.

Заключение

Выбор алгоритма зависит от типа задач, которые предстоит решить, а также от доступных данных и алгоритма, по которому будет построена модель. Некоторые виды задач могут поддерживать различные способы машинного обучения, а некоторые только один.

ЛИТЕРАТУРА

1. Айвазян, С.А. Прикладная статистика: основы моделирования и первичная обработка данных / С.А. Айвазян, И.С. Енюков, Л.Д. Мешалкин. – М. : Финансы и статистика, 1983.
2. Айвазян, С.А. Прикладная статистика: исследование зависимостей / С.А. Айвазян, И.С. Енюков, Л.Д. Мешалкин. – М. : Финансы и статистика, 1985.
3. Прикладная статистика: классификация и снижение размерности / С.А. Айвазян [и др.] – М. : Финансы и статистика, 1989.
4. Вапник, В.Н. Восстановление зависимостей по эмпирическим данным / В.Н. Вапник. – М. : Наука, 1979.

УДК 004.8

ЗАДАЧИ, РЕШАЕМЫЕ С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

М.А. ШЕВЦОВ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматриваются задачи, связанные с машинным обучением, которые можно разделить на то, какой вид прогнозирования или предсказания они будут совершать. В этой статье мы рассмотрим основные виды задач, решаемые с помощью машинного обучения.

1. Задача классификации (англ. **classificationproblem**). Имеется множество объектов (ситуаций), разделенных некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

В математической статистике задачи классификации называются также задачами дискриминантного анализа. В машинном обучении задача классификации решается, в частности, с помощью методов искусственных нейронных сетей при постановке эксперимента в виде обучения с учителем.

Типы классов:

- *двухклассовая классификация*. Наиболее простой в техническом отношении случай, который служит основой для решения более сложных задач;
- *многоклассовая классификация*. Когда число классов достигает многих тысяч, задача классификации становится существенно более трудной;
- *непересекающиеся классы*;
- *пересекающиеся классы*. Объект может относиться одновременно к нескольким классам;
- *нечеткие классы*. Требуется определять степень принадлежности объекта каждому из классов, обычно это действительное число от 0 до 1.

Основные задачи:

- определение принадлежности объекта к определенному классу.

2. Кластерный анализ (англ. **clusteranalysis**) – процедура, выполняющая сбор данных, содержащих информацию о выборке объектов, и затем упорядочивающая объекты в однородные группы. Задача кластеризации относится к статистической обработке, а также к широкому классу задач обучения без учителя.

Независимо от предмета изучения применение кластерного анализа предполагает следующие этапы:

- отбор выборки для кластеризации. Подразумевается, что имеет смысл кластеризовать только количественные данные;
- определение множества переменных, по которым будут оцениваться объекты в выборке, то есть признакового пространства;
- вычисление значений той или иной меры сходства (или различия) между объектами;
- применение метода кластерного анализа для создания групп сходных объектов;
- проверка достоверности результатов кластерного решения.

Основные задачи:

- разработка типологии или классификации;
- исследование полезных концептуальных схем группирования объектов;
- порождение гипотез на основе исследования данных;
- проверка гипотез или исследования для определения, действительно ли типы (группы), выделенные тем или иным способом, присутствуют в имеющихся данных.

3. Регрессионный анализ (англ. **regressionanalysis**) – метод моделирования измеряемых данных и исследования их свойств. Данные состоят из пар значений зависимой переменной (переменной отклика) и независимой переменной (объясняющей переменной).

Основные задачи:

- определение степени детерминированности вариации критериальной (зависимой) переменной предикторами (независимыми переменными);
- предсказание значения зависимой переменной с помощью независимой (независимых);

- определение вклада отдельных независимых переменных в вариацию зависимой.

Регрессионный анализ нельзя использовать для определения наличия связи между переменными, поскольку наличие такой связи и есть предпосылка для применения анализа.

Заключение

Каждый тип задач классифицируется по типу решаемых задач, сферы применения и алгоритмам которые поддерживают решение данной проблемы, но важно помнить, что многие задачи разделены весьма условной линией и могут быть взаимозаменяемы в определенных условиях.

ЛИТЕРАТУРА

1. Дрейпер, Н. Прикладной регрессионный анализ. Множественная регрессия = AppliedRegressionAnalysis / Н. Дрейпер, Г. Смит . – 3-е изд. – М. : Диалектика, 2007. – 912 с.
2. Шлезингер, М. Десять лекций по статистическому и структурному распознаванию / М. Шлезингер, В. Главач. – Киев : Наукова думка, 2004.
3. Mitchell, T. Machine Learning / T. Mitchell. – McGraw-Hill Science/Engineering/Math, 1997.
4. Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell (1983), Machine Learning: An Artificial Intelligence Approach, Tioga Publishing Company.

УДК 004.588

**АНАЛИЗ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ,
ПРЕДНАЗНАЧЕННЫХ ДЛЯ ИЗУЧЕНИЯ АНГЛИЙСКОГО ЯЗЫКА****Н.В. ЮРЦЕВИЧ****(Представлено: Т.С. РУДЬКОВА)**

Рассматриваются функциональные возможности наиболее популярных мобильных приложений, предназначенных для изучения английского языка. Приводится обоснование необходимости создания нового мобильного приложения, которое поможет решить актуальную проблему изучения технического английского языка людям, интересующимся сферой информационных технологий.

На сегодняшний день в мире насчитывается более двухсот государств, а языков, на которых говорят их жители, – несколько тысяч. Международные языки позволяют людям общаться друг с другом, независимо от национальности и места проживания. Английский язык является одним из международных языков. Значение английского языка в современном мире настолько велико, что его знание не является привилегией и роскошью. Его учат все и повсеместно: в школах, университетах, на курсах. В век цифровых технологий любой желающий может изучать английский язык с помощью специализированных мобильных приложений. Подразумевается, что любой образованный человек просто обязан владеть английским языком, так как именно он является его ключом к дальнейшему самообразованию и самосовершенствованию. Именно поэтому сейчас существует так много мобильных приложений, предлагающих научить пользователей английскому.

В сфере информационных технологий английский язык является основным языком отрасли. Большинство технической документации и специализированной литературы написано на английском языке. Кроме того, 90% заказчиков программного обеспечения – это компании из Западной Европы или США, и языком коммуникации с ними является английский. Исходя из этих данных можно сделать вывод, что знание английского языка в сфере информационных технологий важно для успешной работы специалиста.

В последние годы сфера мобильных приложений стремительно развивается. Это обусловлено тем, что у подавляющего большинства людей есть современный мобильный телефон, и люди активно устанавливают мобильные приложения. Несомненным плюсом мобильных приложений является то, что мобильный телефон компактный, благодаря этому пользователь может использовать приложение практически везде, где он находится. Исходя из всего этого, объектом исследования в данной статье стали приложения для изучения английского языка пользователями, интересующимися сферой информационных технологий.

На сегодняшний день довольно много приложений, которые бы позволили изучать английский язык, но приложения для специалистов в сфере информационных технологий отсутствуют. Ниже будут рассмотрены популярные бесплатные аналоги с сервиса Google Play [1]. Первым примером рассмотренного приложения является «Английский разговорник», пользовательский интерфейс которого представлен на рисунке 1.

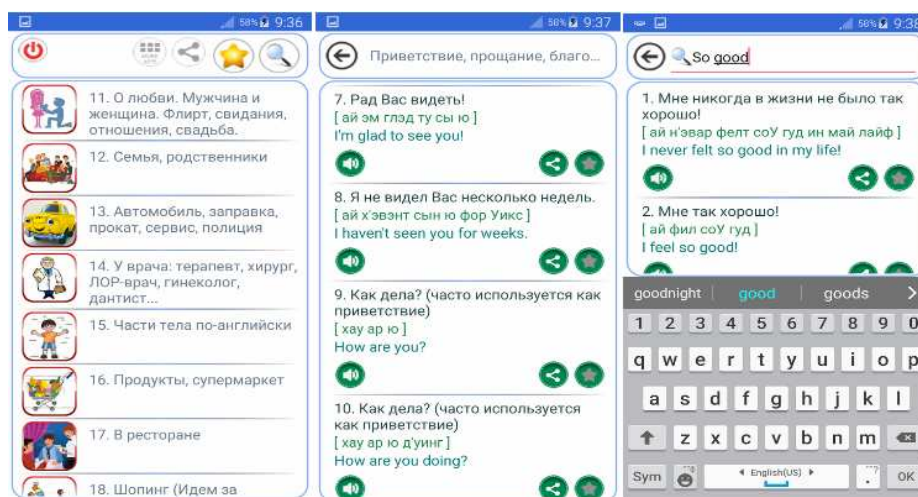


Рисунок 1. – Пользовательский интерфейс мобильного приложения «Английский разговорник»

В приложении представлено 34 категории слов, каждая категория содержит в себя фразы и слова с транскрипцией и произношением – пользователь имеет возможность сохранить слова в категорию избранное, а также реализован поиск по всем категориям. Основное преимущество приложения – не требует подключения к интернету. Также приложение имеет удобный интерфейс и хорошую базу слов. Благодаря этому приложение пользуется популярностью у пользователей (более 100 тыс. установок) и имеет высокий рейтинг (4,7 звезды). Основной недостаток приложения – русскоязычная транскрипция [2].

Вторым примером является приложение «Выучи 90% слов за неделю!», представляющее набор тестов для изучения 300 самых употребляемых английских слов. Также пользователь имеет возможность просмотра статистики. Пример пользовательского интерфейса представлен на рисунке 2.



Рисунок 2. – Пользовательский интерфейс приложения «Выучи 90% слов за неделю!»

Основное преимущество приложения – оно не требует подключения к интернету. Также приложение имеет удобный интерфейс и содержит действительно необходимые слова. Благодаря этому приложение пользуется популярностью у пользователей (более 500 тысяч установок) и имеет высокий рейтинг (4,2 звезды). Основной недостаток приложения – отсутствие категорий [3].

Рассмотренные аналоги не в полной мере отвечают требованиям пользователей, однако при проектировании мобильного приложения для изучения английского языка IT-специалистами можно использовать лучшие практики этих приложений и улучшить их уникальными функциями, которые будут интересны и полезны потенциальным пользователям.

Исходя из рассмотренных выше примеров мобильных приложений, существует необходимость в реализации мобильного приложения со следующим функционалом:

- изучение часто встречающихся слов в технической документации на английском языке;
- выполнение тестовых заданий;
- просмотр статистики (количество изученных слов и категорий, результаты тестов);
- создание, редактирование и удаление пользовательских категорий и прохождение тестов по данным категориям;
- выбор оформления приложения пользователем.

ЛИТЕРАТУРА

1. Google Play // Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Google_Play. – Дата доступа: 11.09.2017.
2. Английский разговорник – Google Play [Электронный ресурс]. – Режим доступа: <https://play.google.com/store/apps/details?id=com.saidjon.ssmphrasebookruen>. – Дата доступа: 12.09.2017.
3. Выучи 90% слов за неделю! – Google Play [Электронный ресурс]. – Режим доступа: <https://play.google.com/store/apps/details?id=com.alexuvarov.android.learntop300englishwords>. – Дата доступа: 12.09.2017.

УДК 004.588

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ
ДЛЯ ИЗУЧЕНИЯ АНГЛИЙСКОГО ЯЗЫКА «HELLO-IT-WORLD»

Н.В. ЮРЦЕВИЧ

(Представлено: Т.С. РУДЬКОВА)

Рассматриваются вопросы разработки мобильного приложения, предназначенного для изучения технического английского языка в сфере информационных технологий.

В мире существует множество стран, а языков, на которых говорят их жители, еще больше. И в данном случае речь идет не о сотне-другой, а о нескольких тысячах. Возникает вопрос, как населению Земли понять друг друга? Этой цели и служат международные языки, которые позволяют всем нам общаться друг с другом, независимо от национальности и места проживания. Один из них – английский язык. Более того, английский язык – язык межнационального общения для всего человечества. Поэтому в настоящее время актуальной является разработка мобильного приложения «Hello-IT-World», которое должно удовлетворять следующим требованиям:

- наличие простого и удобного пользовательского интерфейса;
- наличие обширной базы слов и категорий по IT-тематике;
- наличие тестовых заданий и статистики для каждой категории слов;
- наличие широких функциональных возможностей, а именно: создания и редактирование новых категорий, добавление и редактирование слов в созданные категории, добавление тестовых заданий и статистики по созданным категориям.

При разработке гибридного мобильного приложения использовалась платформа Apache Cordova. Она позволила использовать стандартные веб-технологии, такие как HTML, CSS и JavaScript для кросс-платформенной разработки, избегая родного языка разработки для каждой из мобильных платформ.

HTML был использован для создания каркаса приложения. HTML (от англ. HyperText Markup Language – «язык гипертекстовой разметки») – стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства [1].

CSS был использован для оформления внешнего вида приложения. CSS (англ. Cascading Style Sheets – каскадные таблицы стилей) – формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL [2].

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный и функциональный стили. Он был использован для написания функциональной части приложения [3].

Web SQL Database использовался для доступа к индексируемым базам данных.

Web SQL Database API не является частью спецификации HTML5, это отдельная спецификация, которая вводит набор программных интерфейсов для управления клиентской базой данных с помощью языка SQL. Существуют три основных метода, определенных в данной спецификации:

- 1) OpenDatabase – метод создает объект базы данных, либо использует существующую базу данных;
- 2) Transaction – этот метод дает нам возможность контролировать действия выполнения или отката запроса в зависимости от ситуации;
- 3) ExecuteSQL – метод используется для выполнения фактического SQL запроса [4].

При разработке программного обеспечения разработана функциональная структура, которая состоит из следующих подсистем:

- *подсистема изучения слов* – отвечает за корректное отображение категорий и слов, извлекаемых из базы данных приложения, предназначена для изучения пользователями написания и произношения английских слов;
- *подсистема прохождения тестов* – обеспечивает генерацию и корректное отображение тестовых заданий для каждой категории слов с возможностью выбора пользователем направления перевода;
- *подсистема сбора статистики* – предоставляет возможность сбора и отображения статистической информации;
- *подсистема поиска слов* – предоставляет пользователю возможность поиска слов по базе данных приложения;

- подсистема добавления категорий и слов – позволяет пользователю приложения создавать новые категории и добавлять в них английские слова с переводом, а также редактировать и удалять их;
- подсистема настройки приложения – позволяет пользователю настраивать внешний вид мобильного приложения.

Функциональная структура системы представлена на рисунке 1.

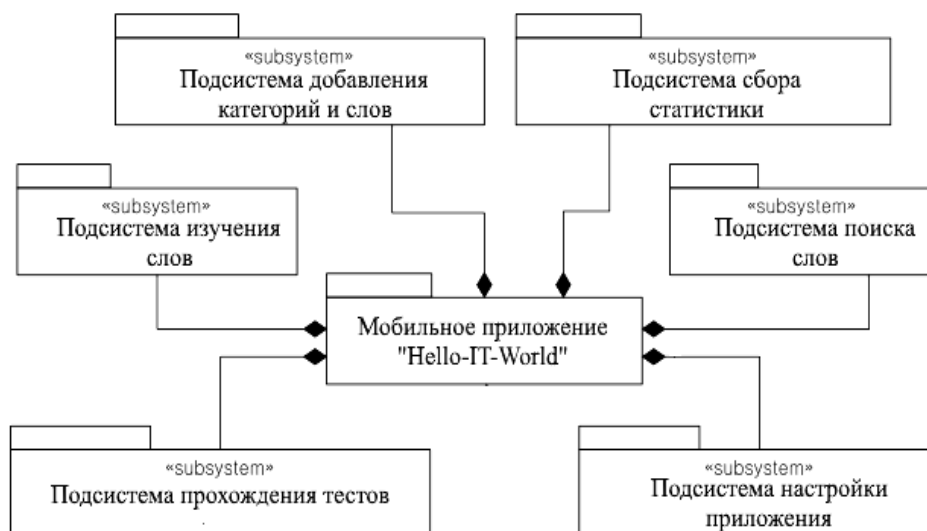


Рисунок 1. – Функциональная структура приложения

Подсистема прохождения тестов состоит трех частей, таких как:

- выбор категории и направления перевода, за что отвечает функция `initCategoryList`;
- автоматически генерируемый тест, за генерацию которого отвечает функция `startTest`;
- окна с результатами прохождения теста, за отображение которого отвечает функция `rez`.

При нажатии на выпадающий список пользователь может выбрать желаемую категорию для прохождения теста. Также с помощью флажков, пользователь может выбрать направления перевода слов. После нажатия кнопки «Начать тест», тест будет автоматически сгенерирован.

В автоматически сгенерированном тесте учитывается выбор категории и направление перевода, указанные пользователем ранее. Количество вопросов в тесте равно количеству слов в категории. В каждом вопросе есть четыре варианта ответа, и только один правильный. Основное преимущество генерируемого теста в том, что при каждом новом запуске теста варианты ответа расположены в абсолютно случайном порядке, и при этом в одном вопросе не может быть двух одинаковых вариантов ответа.

При прохождении теста пользователь в любой момент может вернуться к настройкам теста, благодаря кнопке «Назад» (функция `onStart`). Для пролистывания теста в нижней части экрана появляется кнопка «Вверх» (функция `scroll`), при нажатии на которую пользователь в любой момент может вернуться к началу теста. Выбор ответа осуществляется нажатием на флажок напротив вариантов ответа. Для удобства пользователя, на тот случай, если он пропустит несколько вопросов при прохождении теста, по умолчанию флажок установлен на нулевой (скрытый) вариант ответа.

После ответа на все вопросы пользователь должен нажать кнопку «Результаты теста» тест закроется, и пользователь увидит статистику своих ответов (рис. 2).

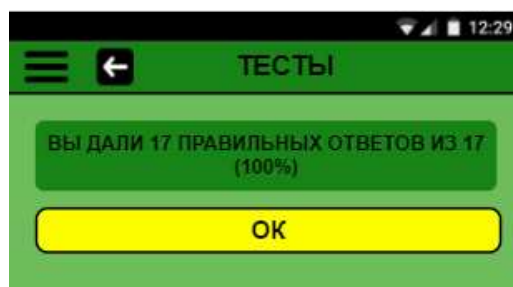


Рисунок 2. – Результаты прохождения теста

Результаты прохождения теста влияют на рейтинг слов. Изначально слово имеет рейтинг 0, при правильном ответе на вопрос теста рейтинг увеличивается на 1, при неправильном ответе – уменьшается на 1. Также процент правильных ответов влияет на рейтинг категории. Изначально категория имеет рейтинг 0. Если пользователь дает от 68 до 100% правильных ответов, рейтинг увеличивается на 1, если правильных ответов от 34 до 67% – рейтинг категории не меняется, если же пользователь дает менее 34% правильных ответов, рейтинг категории уменьшается на 1. Границы рейтинга слов и категорий от 0 до 5. Графически рейтинги отображаются около слов и категорий в словаре и статистике с помощью звезд, число желтых звезд численно равно рейтингу слова или категории.

После того как был написан функционал мобильного приложения, оно было протестировано, чтобы удостовериться в том, что программный продукт соответствует указанным ранее требованиям. Тесты выявили несколько дефектов приложения, которые были незамедлительно устранены. Повторное тестирование дефектов не выявило.

Результатом проделанной работы стало мобильное приложение для изучения английского языка. Разработанный программный продукт может использоваться для изучения английского языка пользователями, которые планируют приобретать и совершенствовать навыки английского языка для последующего применения полученных знаний в сфере информационных технологий.

ЛИТЕРАТУРА

1. HTML [Электронный ресурс] // Википедия, свободная энциклопедия. – Электронный ресурс. – Режим доступа: <https://ru.wikipedia.org/wiki/HTML>. – Дата доступа: 23.09.2017.
2. CSS [Электронный ресурс] // Википедия : свободная энцикл. – Электронный ресурс. – Режим доступа: <https://ru.wikipedia.org/wiki/CSS>. – Дата доступа: 23.09.2017.
3. JavaScript [Электронный ресурс] // Википедия, свободная энцикл. – Электронный ресурс. – Режим доступа: <https://ru.wikipedia.org/wiki/JavaScript>. – Дата доступа: 23.09.2017.
4. HTML5 Web SQL Database [Электронный ресурс] // Web онлайн учебники. – Электронный ресурс. – Режим доступа: <http://www.w3big.com/ru/html/html5-web-sql.html>. – Дата доступа: 24.09.2017.

УДК 004.588

ИСПОЛЬЗОВАНИЕ APACHE CORDOVA КАК ПЛАТФОРМЫ ДЛЯ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ В СВЯЗКЕ С ФРЕЙМВОРКОМ IONIC

Д.В. ДРОЗД**(Представлено: Д.В. ПЯТКИН)**

Рассматривается использование Apache Cordova и Ionic веб-разработчиками для нескольких платформ с единой реализацией, позволяющей использовать стандартные веб-технологии, такие как HTML5, CSS3 и JavaScript, для кроссплатформенной разработки, избегая родного языка разработки для каждой из мобильных платформ.

При разработке мобильного приложения требуется отдельная реализация для каждой мобильной платформы. В связи с этим усложняется разработка мобильного приложения, привлекая для проекта большое количество разработчиков. Использование *Apache Cordova* позволяет свести штат разработчиков к минимуму, минимизировать финансовые и временные затраты на разработку и позволяет включить в разработку мобильного приложения веб-разработчиков. В Фреймворк *Ionic* входит набор инструментов и виджетов для создания приложений. Это анимированные всплывающие окна на странице, кнопки, слайдеры и многое другое.

Данная работа поможет получить представление о платформе *Apache Cordova* и Фреймворке *Ionic*, что хорошо подходит для кроссплатформенной разработки мобильных приложений.

Apache Cordova – это платформа разработки мобильных приложений с открытым исходным кодом, которая позволяет использовать стандартные веб-технологии, такие как HTML5, CSS3 и JavaScript для кроссплатформенной разработки, избегая родного языка разработки для каждой из мобильных платформ. Приложения выполняются внутри обертки нацеленной на каждую платформу и полагаются на стандартные API для доступа к датчикам устройства, данным и состоянию сети.

На сегодняшний день самой популярной оболочкой для гибридных приложений является *Apache Cordova* (ранее – *PhoneGap*). Это платформа, которая помимо оборачивания кода в *WebView* предоставляет огромный набор плагинов js API, с помощью которых можно полноценно взаимодействовать с операционной системой и устройствами смартфона. У *Apache Cordova* большое сообщество, множество готовых решений и плагинов, но для полноценного приложения не хватает красивого, современного UI, который бы придал приложению удобство управления и ощущение, что это приложение, а не адаптированный сайт.

Ionic – это Фреймворк, с хорошо проработанным графическим интерфейсом, стилизованный под различные платформы для максимальной схожести с нативными приложениями. За основу у Фреймворка взяты *AngularJS* и *SASS*, что принесло ему большую популярность у разработчиков, так как эти технологии сводят порог вхождения к минимуму. Ядром фреймворка является *Apache Cordova*, поэтому все плагины *Apache Cordova* будут отлично работать на *Ionic*.

Чтобы начать работать с *Apache Cordova* и *Ionic* требуется установить несколько настроек для библиотеки и Фреймворка:

- 1) установка пакетного менеджера *node.js* – *npm*. С его помощью можно управлять модулями и зависимостями;
- 2) установка *Node.js* – программная платформа, превращающая JavaScript из узко специализированного языка в язык общего назначения;
- 3) установка *git* клиента;
- 4) установка модуля *cordova* с помощью инструмента *npm* (листинг 1);
- 5) установка модуля *ionic* с помощью инструмента *npm* (листинг 2);
- 6) создание приложения через *ionic* фреймворк (листинг 3);
- 7) добавить платформы в *Apache Cordova*, для которых требуется разработка мобильного приложения (листинг 4);
- 8) борка исходного файла для мобильного приложения (листинг 5).

Листинг 1 – Установка *Apache Cordova*

```
npm install -g cordova
```

Листинг 2 – Установка Фреймворка ionic

```
npm install -g ionic
```

Листинг 3 – Создание нового мобильного приложения

```
ionic start JoinUs
```

Листинг 4 – Добавление платформы android в Apache Cordova

```
cordova platform add android
```

Листинг 5 – Сборка apk мобильного приложения для android устройств

```
ionic cordova build – release android
```

Заключение

В данной работе рассмотрены возможности создания и экспорта веб-приложения в мобильные приложения с использованием платформы *Apache Cordova* и *Ionic* Фреймворк.

Показан базовый пример по экспорту веб-приложения в мобильные устройства на операционной системе *Android*, демонстрирующий простоту применения данных технологий в рамках кроссплатформенной разработки.

ЛИТЕРАТУРА

1. Apache Cordova [Электронный ресурс] Интерфейс командной строки. – Режим доступа: <https://cordova.apache.org/docs/ru/latest/guide/cli/>. – Дата доступа: 25.09.2017.
2. Ionic Framework [Электронный ресурс] Publishing your app. – Режим доступа: <http://ionicframework.com/docs/v1/guide/publishing.html>. – Дата доступа: 25.09.2017.
3. Блог Front-End Developer'a [Электронный ресурс] Гибридные приложения, разработка с Ionic. – Режим доступа: <https://golosay.net/hybrid-apps-developing/>. – Дата доступа: 25.09.2017.

УДК 004.588

**РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ «JOINUS»,
ПРЕДНАЗНАЧЕННОГО ДЛЯ ОРГАНИЗАЦИИ РАЗВЛЕКАТЕЛЬНЫХ
И КУЛЬТУРНО-ОБЩЕСТВЕННЫХ МЕРОПРИЯТИЙ**

Д.В. ДРОЗД

(Представлено: Д.В. ПЯТКИН)

Рассматривается мобильное приложение, предоставляющее возможность пользователям создавать свои объявления о мероприятиях, а также найти пользователю мероприятие по интересу. Представлена разработка приложения «JoinUs».

Современный смартфон или планшет способен полностью заменить весь функционал домашних компьютеров, в связи с чем актуальность приобретает разработка приложений не столько для персональных компьютеров, как для планшетов и смартфонов. С популяризацией мобильных устройств пользователи чаще ищут необходимую информацию на своих мобильных устройствах. Вследствие этого актуальность разработки мобильного приложения возрастает, а вместе повышается актуальность в поисках или организации мероприятий прямо с мобильного устройства.

Разработка данного мобильного приложения «JoinUs», предназначенного для организации развлекательных и культурно-общественных мероприятий, является реализацией мобильного приложения, предоставляющего возможность пользователям создавать свои объявления о мероприятиях, а также найти пользователю мероприятие по его интересу.

Приложение не только должно найти заинтересовавшее пользователя мероприятие, но и предоставить ему всю необходимую информацию об этом событии: дату, место, описание и многое другое. Ознакомительная информация о мероприятии должна быть представлена в простой и понятной форме в виде текста и местоположения на карте. Процесс загрузки, установки, использование должен быть максимально удобен.

Целевой аудиторией мобильного приложения являются любые пользователи, проживающие на территории Республики Беларусь.

Для разработки данного мобильного приложения используется библиотека *Apache Cordova* с *Ionic* фреймворком, данная связка позволяет разработать кроссплатформенное мобильное приложение.

Для дизайна мобильного приложения используется *Material Design*. *Material Design* – дизайн программного обеспечения и приложений от компании Google. Идея дизайна заключается в приложениях, которые открываются и сворачиваются как карточки, используя эффекты теней.

Разработка данного мобильного приложения написано на языке программирования JavaScript с использованием AngularJS Framework, а также современных средств для работы с веб, такими как HTML5, CSS3, JavaScript, Google Maps API.

В качестве сервера выступает *Cloud9*. *Cloud9* – интегрированная среда разработки, предоставляемая по модели облачных вычислений. Среда поддерживает несколько языков программирования, при этом, основной фокус направлен на JavaScript, HTML и CSS. Среда разработки полностью написана на JavaScript и использует Node.js на серверной стороне. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API.

Вся информация на сервере хранится в *MongoDB*. *MongoDB* – документоориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы. Пример данных в MongoDB в виде JSON-подобных документов представлен в листинге 1.

Листинг 1 – Описание данных в MongoDB о пользователе и мероприятии

```
_id: "58b97f69a20c1217adfd9a4f",  
username: "deniskogo",  
firstname: "Denis",  
lastname: "Droz",  
avauser: "data:image/png;base64..."
```

```
_id: "595485da9becc1068b93b546",  
author: "58b97f69a20c1217adfd9a4f",  
header: "Пробное",  
text: "Немного текста",  
date: "2017-06-29T21:00:00.000Z",  
dateCreated: "2017-08-29T09:45:11.188Z",  
x: 55.49285609375866,  
y: 28.77653863769524,  
likeUsers:  
[  
    "58f8764b8112a10877a010ef",  
    "58c0ffda03e8990c585d0127"  
]
```

Мобильное приложение «JoinUs» состоит из множества страниц, каждая из них отвечает за свою задачу:

1. Страница аутентификации пользователя.
2. Страница регистрации пользователя.
3. Персональная страница пользователя.
4. Боковое меню.
5. Список мероприятий.
6. Страница сообщений пользователя.

Заключение

В результате выполненной работы реализовано мобильное приложение «JoinUs» предназначенное для организации развлекательных и культурно-общественных мероприятий. Для мобильного приложения разработан удобный для работы интерфейс, который не требует дополнительного обучения и интуитивно понятен, а также был выделен сервер для хранения необходимой информации для мобильного приложения.

ЛИТЕРАТУРА

1. Магазин мобильной электроники НОУ-ХАУ [Электронный ресурс] Нужен ли планшет в «мире смартфонов». Актуальность планшетов в 2017. 19 июня 2017. Новости. Обзоры и статьи о технике. – Режим доступа: <http://nou-hau.pf/article/aktualnost-planshetov-v-2017>. – Дата доступа: 25.09.17.
2. Википедия [Электронный ресурс] AngularJS. – Режим доступа: <https://ru.wikipedia.org/wiki/AngularJS>. – Дата доступа: 25.09.17.
3. Википедия [Электронный ресурс] Material Design. – Режим доступа: https://ru.wikipedia.org/wiki/Material_Design. – Дата доступа: 25.09.2017.
4. Apache Cordova [Электронный ресурс] Интерфейс командной строки. – Режим доступа: <https://cordova.apache.org/docs/ru/latest/guide/cli/>. – Дата доступа: 25.09.2017.
5. Ionic Framework [Электронный ресурс] Publishing your app. – Режим доступа: <http://ionicframework.com/docs/v1/guide/publishing.html>. – Дата доступа: 25.09.2017.
6. Learn Javascript [Электронный ресурс] Скринкаст по Node.JS. – Режим доступа: <https://learn.javascript.ru/screencast/nodejs>. – Дата доступа: 25.09.2017.
7. Википедия [Электронный ресурс] Cloud9. – Режим доступа: <https://ru.wikipedia.org/wiki/Cloud9>. – Дата доступа: 25.09.2017.

УДК 004.45

**АНАЛИЗ ФУНКЦИОНАЛЬНОСТИ ЭЛЕКТРОННЫХ ПУТЕВОДИТЕЛЕЙ
НА ПРИМЕРЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ И WEB-САЙТА****А.В. ДЫСИН****(Представлено: Д.В. ПОПКОВА)**

Рассмотрено понятие «путеводитель», виды путеводителей. Изучены функции путеводителей. Проведено исследование характеристик web-приложения и web-сайта с точки зрения функциональности. Выделены плюсы и минусы функциональных возможностей программных средств.

Путеводитель – краткое справочное издание. Путеводители по стране, городу и т.п., предназначенные главным образом для туристов, содержат сведения о местных достопримечательностях, учреждениях культуры, путях сообщения, отелях и другую информацию. Широко известны «путеводители по странам – Бедекеры». Путеводители по отдельным учреждениям культуры (музеям, крупным библиотекам) или исторически-художественным комплексам сообщают об их истории, структуре, фондах.

Существует несколько функций путеводителей, которые позволяют раскрыть их сущность в современном мире.

Путеводитель как вид справочного издания

Путеводитель как справочник представляет собой сложно организованную, эклектичную структуру, в которой соединена научно-популярная информация комплексного характера и информация бытового плана, не претендующая на научность и строгость. В зависимости от структуры и содержания текста, формируются различные виды «взаимодействия» читателя с текстом: предварительное и выборочное ознакомление; последовательное, «академическое» изучение; прерывистое чтение, совмещенное с перемещением в пространстве и осмотром достопримечательностей [1].

Трансляция научного знания на уровень обывденного сознания. Путеводитель можно рассматривать как текст, соединяющий научно-справочный и научно-популярный подстили. Будучи текстом, созданным для «культурной рекреации», Бедекер, по сравнению с другими разновидностями научно-популярного жанра, значительно упрощает, редуцирует, унифицирует и эмоционально окрашивает сообщаемую читателю информацию. В наибольшей степени это проявляется в характере введения в текст специальных терминов, в особенностях цитирования, в расширенном использовании эпитетов, гипербола и метафор.

Путеводитель как руководство по формированию новых поведенческих стереотипов. В самом жанре путеводителя, рожденного культурой, как ответ на массовый запрос в руководстве, априорно заложена функция воздействия на читателя. Используемые приемы руководства достаточно разнообразны: от демонстрации престижности тех или иных действий до советов, подбадриваний, понуждений и даже императивов. Путеводитель контролирует также визуальное восприятие: не только отбирая объекты или степень детализации при осмотре, но и предписывая «соответствующие» реакции и впечатления. Текст диктует распределение времени, характер, скорость, ритм передвижения человека в туристском пространстве [1].

Рекламно-имиджевые функции путеводителя. В туристском пространстве регион выступает своеобразным «товаром» и задача путеводителя – привлечь к нему внимание. Основные приемы, используемые для реализации данной функции – позиционирование, персонификация имиджа, визуализация, преувеличение, сопоставление с всемирно известными достопримечательностями и т.д.

Путеводитель как инструмент конструирования достопримечательностей

Конструирование достопримечательностей включает ряд операций: переведение памятника из поля функционирования искусства в поле функционирования массовой культуры; упрощение, а порой и искажение смыслов, ценностей, значений; тиражирование и организация массового потребления; переведение из сферы незаинтересованного эстетического восприятия в сферу престижного потребления [1].

Электронный путеводитель – широкое понятие.

Он может иметь разные формы:

1. *Аудиогид* – воспроизводится на любом устройстве, поддерживающем mp3. *Аудиоэкскурсия* – это маршрут, состоящий из остановок, каждой из которых соответствует один трек.
2. *Электронный буклет* – на экране воспроизводится файл буклета, с картой и текстом (обычно в pdf-формате).
3. *Интерактивный справочник* – на экране воспроизводится интерфейс программы с доступом к карте и справочной информации.
4. *Мультимедийный путеводитель* – на карте размещены гипертекстовые ссылки, открывающие фотографии, описание, звуковой комментарий, видеоролик.

5. *Мультимедийный навигатор* (GPS-путеводитель) – определяет местонахождение экскурсанта, показывает дорогу и автоматически включает звуковой комментарий [3].

Следует различать так называемые «умные путеводители», которые сами могут определить, где находится человек, и путеводители с ручным управлением.

Под открытым небом используют путеводители, имеющие спутниковую навигацию. Сейчас действует американская система GPS и российская ГЛОНАСС. Однако сила сигнала спутниковой навигации очень мала – в помещениях GPS не действует.

Задача автоматического ориентирования внутри помещений решается одним из *методов альтернативной навигации*:

1. *Беспроводные сети (WiFi)*. Принцип работы: в здании устанавливается сервер и беспроводные точки доступа (WirelessAccessPoint). Программа считывает через точки доступа информацию, хранящуюся в базе данных сервера. Плюсы – не надо хранить много информации, работает быстрее; базу данных можно обновлять централизованно. Минусы – дорого, сложная технология.

2. *Навигация по мобильной связи (GSM)*. Местоположение определяется по близости к базовой станции сотовой связи.

3. *Радиочастотная идентификация (RFID)*. Навигация ведется по специальным RFID-меткам. Это чипы, которые прикрепляются к объектам. Приемник считывает информацию с этих чипов. Такая система активно используется в логистике, на больших складах, в транспорте.

Очевидно, такие проекты требуют немалых затрат, поэтому создаются путеводители с ручным управлением.

Электронный путеводитель может быть в виде Web-сайта или программы.

Web-сайт – это объединенная под одним адресом (доменным именем) совокупность документов юридического или физического лица. По умолчанию подразумевается, что сайт располагается в сети Интернет, но он также может публиковаться в локальной сети компании – в сети Интранет [1].

Интернет-сайты или онлайн-путеводители обновляются быстрее и чаще, чем их аналоги, поэтому в них можно найти больше заслуживающей доверия информации, чем в книгах самого авторитетного издательства. На таких сайтах размещаются GPS-маршруты экскурсии, информация о достопримечательностях и их фотографии. Достоинством таких путеводителей является бесплатная информация, постоянные обновления, нет необходимости хранить много информации; недостатком – наличие доступа в Интернет, большие объемы загружаемой информации [1].

Программа устанавливается на мобильное устройство (КПК, смартфон, iPod, автонавигатор), персональный компьютер (ноутбук) или запускается с диска. Достоинством является автономность, приспособленность для конкретной цели и устройства. Недостатком – программа обычно платная, данные могут устаревать, не всегда хорошо устанавливается, бывают сбои.

В последнее время популярными становятся информационные сенсорные киоски с электронными системами навигации. Такие справочники-путеводители устанавливаются преимущественно в музеях, торговых, бизнес- и публичных центрах.

Пример мобильного приложения «Redigo». Это – компактный гид по странам мира от Австралии до Японии. Он содержит следующую информацию: общая информация о стране, информация о местах и городах, которые рекомендуется посетить, оффлайн карты, где отмечены достопримечательности, информация об актуальных событиях. На рисунке 1 представлен внешний вид приложения «Redigo».

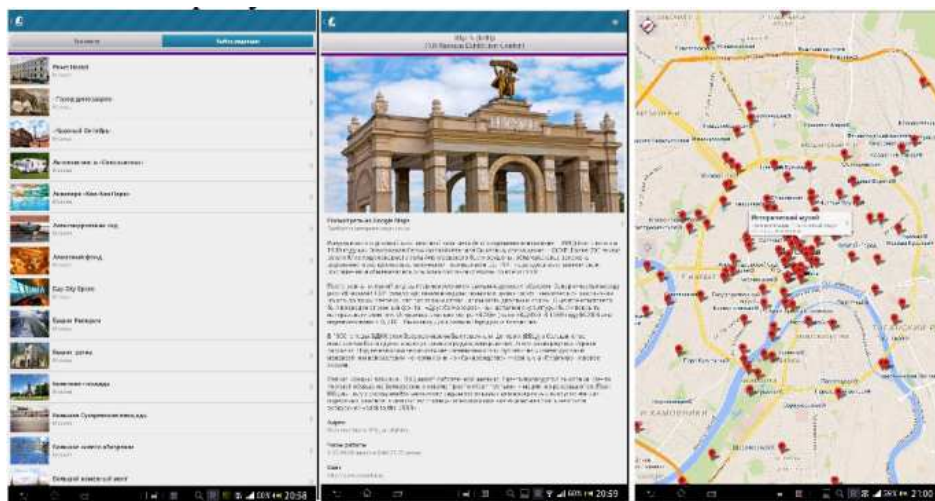


Рисунок 1. – Вид сайта «Redigo»

Приложение Redigo – это гид по большому количеству стран и городов. Информация о каждом городе скачивается на устройство отдельно. В данном примере разберем функциональность приложения на примере города Санкт-Петербурга.

Функциональные возможности приложения Redigo:

- поиск достопримечательностей города;
- поиск достопримечательностей посредством геолокации;
- наличие разговорника на таких языках, как английский, испанский, итальянский, китайский, немецкий и французский;

- наличие разговорника в картинках;

- поиск конкретной достопримечательности по названию;
- возможности скачать оффлайн-карту города либо ее онлайн просмотр.

Однако у данного приложения имеются следующие недоработки:

- несистематизированная информация о достопримечательностях города;
- отсутствие информации об объектах размещения;
- отсутствие информации о предприятиях питания;
- отсутствие возможности проложить маршрут;
- для загрузки оффлайн-карты необходим доступ в интернет.

Примеры Web-сайта – «Путеводитель по Санкт-Петербургу». Это – web-сайт путеводитель по Санкт-Петербургу, который служит помощником не только для туристов, но и для жителей города. На сайте представлена общая информация о городе, о существующих экскурсиях по Санкт-Петербургу и пригородам. В разделе «Карта» отмечены главные достопримечательности города, а также средства размещения, предприятия питания, объекты здравоохранения, объекты досуга и развлечений, транспортного сообщения, последние новости города.

Внешний вид сайта «Путеводитель по Санкт-Петербургу» (<http://cityguidespb.ru>) представлен на рисунке 2.

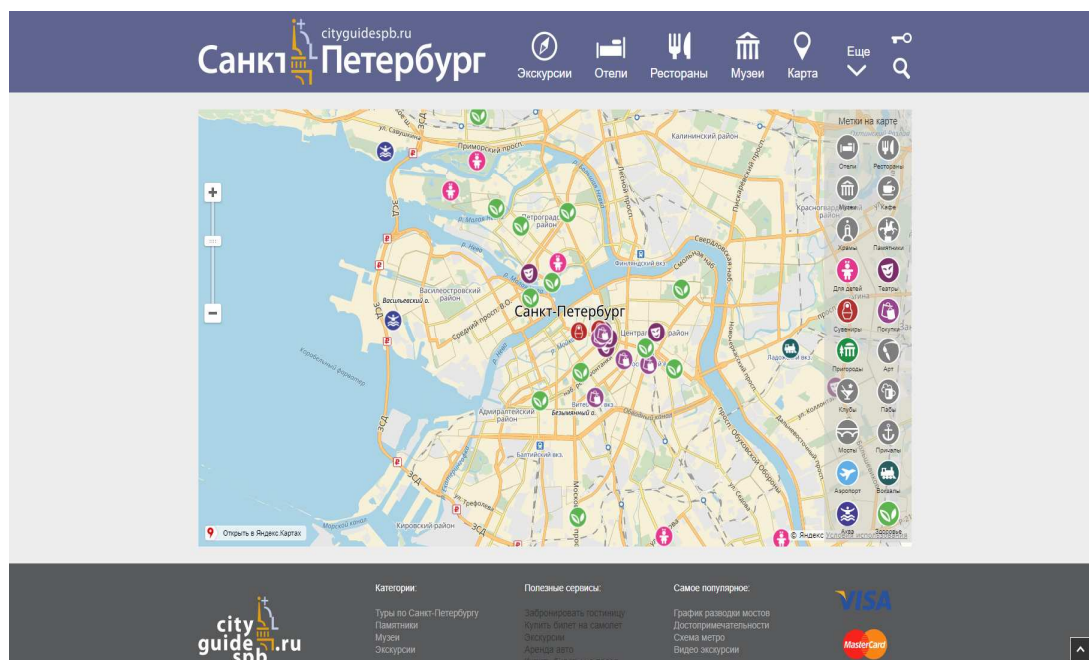


Рисунок 2. – Вид сайта «Путеводитель по Санкт-Петербургу»

Функциональные возможности Web-сайта «Путеводитель по Санкт-Петербургу»:

- предоставление актуальной информации о городе;
- поиск объектов города таких как: объекты размещения, предприятия питания, музеи и т.д.;
- поиск конкретной достопримечательности по названию;
- предоставление информации об экскурсиях и турах Санкт-Петербурга;
- предоставление информации о торговых комплексах города;
- предоставление информации о маршрутах общественного транспорта следующих до необходимого объекта Санкт-Петербурга;

- возможности оставить отзыв;
- реализации личного кабинета пользователя;
- предоставление видео экскурсий.

Недостатки web-сайта:

- для работы с сайтом необходим постоянный доступ к интернету;
- отсутствие возможности проложить маршрут до объекта города на карте;
- отсутствие информации о расстоянии до искомого объекта Санкт-Петербурга;
- большие объемы загружаемой информации.

Таким образом, на основе анализа выбранных источников, можно сделать следующие **выводы:** приложение Redigo как источник информации о странах, за исключением возможности использования-оффлайн, является менее функциональным, так как имеет только общую информацию о достопримечательностях города и страны и не может выступать подробным путеводителем. В то время как Web-сайт «Путеводитель по Санкт-Петербургу» имеет полную информацию обо всех необходимых для путешественника объектах и более полный функционал.

На основе данных выводов выявлена лучшая функциональность у тех видов Web-приложений и Web-сайтов, которые предназначены для узкого направления, так как в них можно наиболее полно отобразить интересующую информацию и сделать расширенные функциональные возможности.

Учитывая все плюсы и минусы рассмотренных в данной работе web-приложения и web-сайта, можно говорить о необходимости разработать улучшенную модель путеводителя, которая будет ориентирована на один город и включать в себя всю необходимую функциональность.

ЛИТЕРАТУРА

1. Сайт – Википедия [Электронный ресурс]. – Режим доступа: <http://ru.wikipedia.org/wiki/Сайт>. – Дата доступа: 10.09.2017.
2. Веб-сайты: определение и применение [Электронный ресурс]. – Режим доступа: <http://www.webtec.com.ua/articles/index/view/web-site/>. – Дата доступа: 13.09.2017.
3. Электронные путеводители: Что это и с чем их едят? [Электронный ресурс]. – Режим доступа: <https://www.skyscanner.ru/news/elektronnye-putevoditeli-chto-eto-i-s-chem-ikh-edyat>. – Дата доступа: 15.09.2017.

УДК004.924

ОСОБЕННОСТИ РЕАЛИЗАЦИИ ИНТЕРАКТИВНОГО 3D-ПУТЕВОДИТЕЛЯ ПО ГОРОДУ ПОЛОЦКУ

А.В. ДЫСИН**(Представлено: Д.В. ПОПКОВА)**

Рассмотрена программная библиотека для языка JavaScriptWebGraphicsLibrary. Выделены ее основные достоинства и недостатки, проанализирована поддержка библиотеки существующими браузерами. Рассмотрены основные библиотеки и фреймворки для разработки на WebGL. Разработан интерактивный 3D-путеводитель по городу Полоцку.

Web-технологии прочно вошли в повседневную жизнь современного человека. Просмотр новостей, совершение покупок, общение и работа в большинстве случаев осуществляются с помощью всемирной сети. Индустрия услуг и развлечений в сети Интернет стремительно развивается, ведущие разработчики программного обеспечения улучшают поддержку трехмерной графики в своих продуктах. Традиционно ее поддержка ограничивалась высокопроизводительными компьютерами или специализированными игровыми консолями, а программирование требовало применения сложных алгоритмов. Однако благодаря росту производительности персональных компьютеров и расширению возможностей браузеров, стало возможным создание и отображение трехмерной графики с применением Web-технологий.

WebGL (WebGraphicsLibrary) – программная библиотека для языка JavaScript, предназначенная для визуализации интерактивной трехмерной графики и двухмерной графики в пределах совместимости Web-браузера без использования плагинов. Одно из преимуществ WebGL – приложения конструируются как Web-страницы, то есть одна и та же программа будет успешно выполняться на самых разных устройствах (к примеру, на смартфонах, планшетных компьютерах и игровых консолях) [1].

С развитием языка гипертекстовой разметки HTML разработчики получили возможность создавать все более сложные Web-приложения. Изначально HTML предлагал только возможность отображения статического контента, но с добавлением поддержки языка JavaScript стало возможным реализовывать более сложные взаимодействия элементов и отображения динамического контента. Внедрение стандарта HTML5 позволило использовать новые возможности, включая поддержку двухмерной графики в виде тега canvas. Создание технологии WebGL позволило отображать и манипулировать трехмерной графикой на Web-страницах с помощью JavaScript. При помощи WebGL разработчики могут создавать совершенно новые пользовательские интерфейсы, трехмерные игры и использовать трехмерную графику для визуализации различной информации. Несмотря на внушительные возможности, WebGL отличается от других технологий доступностью и простотой использования.

В настоящий момент WebGL поддерживается браузерами:

1. *Десктопные браузеры:*
 - MozillaFirefox(с 4-й версии);
 - GoogleChrome (с 9-й версии);
 - Safari (с 6-й версии, по умолчанию поддержка WebGL отключена, чтобы ее включить пользователю необходимо поставить соответствующий флаг в инструментах разработчика);
 - Opera (с 12-й версии, по умолчанию также отключена поддержка WebGL);
 - IE (с 11-й версии, для других версий можно использовать сторонние плагины, например, IEWebGL).

2. *Мобильные браузеры:*

- Android-браузер (поддержка WebGL только на нескольких устройствах);
 - OperaMobile (начиная с 12-й версии, и только для OSAndroid);
 - IOS (полная поддержка с версии 8.1);
 - Firefox for mobile (с 4-й версии);
 - Google Chrome для Android (с 25-й версии) [2].
- Преимущества использования WebGL:
- кроссбраузерность и отсутствие привязки к определенной платформе;
 - использование языка JavaScript, который достаточно распространен;
 - автоматическое управление памятью. В отличие от OpenGL, в WebGL не надо выполнять специальные действия для выделения и очистки памяти;
 - поскольку WebGL для рендеринга графики использует графический процессор на видеокарте (GPU), для этой технологии характерна высокая производительность, которая сравнима с производительностью нативных приложений.

Недостатки использования WebGL:

- отсутствие полной поддержки во всех браузерах;

- медленная работа на мобильных устройствах [3].

Технология WebGL использует низкоуровневое API, что облегчает разработчикам браузеров внедрять ее в свои продукты, но создает достаточно большие трудности при создании интерфейсов. Для упрощения работы разработчиков сайтов были созданы библиотеки, фреймворки и сторонние программные средства.

Библиотеки и фреймворки для разработки на WebGL:

1. *Библиотека WebGLU*. Набор утилит низкого и высокого уровня для разработки приложений на WebGL. WebGLU сконструирован таким образом, что разработчик может сосредоточиться на конечном результате, но библиотека не ограничивает разработчиков, которые хотят больше контроля над системой разработки. Это достигается за счет возможности использования низкоуровневых функций для работы с WebGL API.

2. *Библиотека GLGE*. Библиотека для разработки приложений с использованием WebGL. Библиотека ориентирована больше на динамическое изменение сцены. Однако последние изменения в репозитории осуществлялись в 2014 году, поэтому есть причины усомниться в актуальности библиотеки на сегодняшний день.

3. *Библиотека Three JS*. Наиболее популярная и активно развивающаяся библиотека на сегодняшний день. Подробная и доступная документация и огромное количество рабочих примеров делают эту библиотеку одним из лидеров среди аналогичных систем.

4. *Библиотека Babylon JS*. Библиотека с открытым исходным кодом для создания полноценных 3D-приложений и игр, работающих в веб-браузере без использования сторонних плагинов и расширений. Babylon JS по своим возможностям близок к ThreeJS, однако имеет в своем арсенале некоторые встроенные функции, недоступные в Three JS из коробки. К таким приятным особенностям относятся встроенный физический движок oimo.js – достаточно простой способ создать реалистичный ландшафт, используя карту высот. Разумеется, в threejs также присутствуют такие возможности, но реализованы они при помощи различных дополнительных приложений. Однако за функциональность библиотеки приходится платить нескромным весом в 800 кб.

На основе данного исследования был разработан интерактивный 3D-путеводитель по городу Полоцку с использованием технологии WebGL.

3D-путеводитель состоит из двух функциональных частей: основной клиентской части сайта и его административной панели. Перед полной загрузкой главной страницы отображается анимированный предзагрузчик. Главная страница клиентской части сайта представлена на рисунке 1.



Рисунок 1. – Главная форма интерактивного путеводителя

При выборе необходимого района осуществляется анимированный переход к выбранной локации с отображением всех существующих объектов города. На странице справа находится поле для поиска предприятий размещения и питания, достопримечательностей, а также торговых помещений с сувенира-

ми. Отображение результатов поиска можно выбрать как по алфавитному порядку, так и по классификации, представленной на рисунке выше. Страница отдельного района Полоцка представлена на рисунке 2.

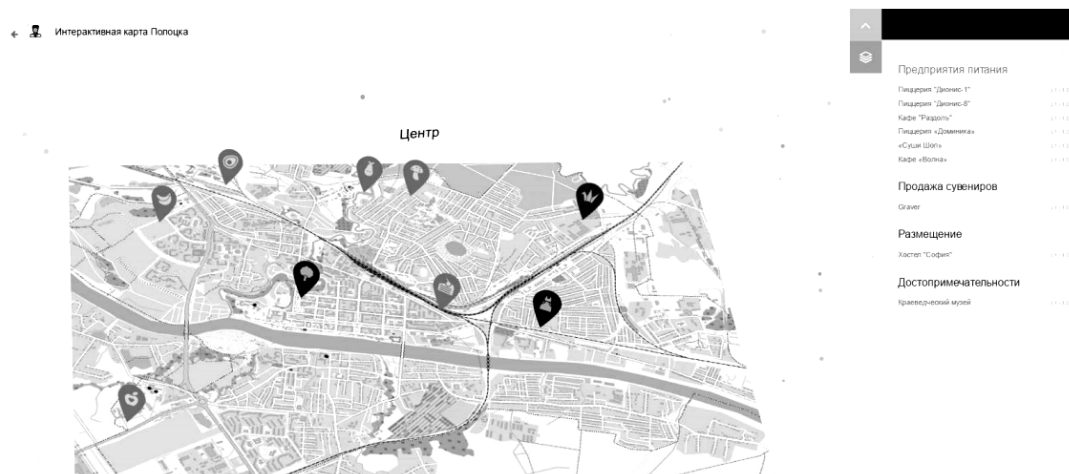


Рисунок 2. – Страница отдельного района Полоцка

На следующем шаге пользователю предоставляется блок с полной информацией о выбранном объекте. При нажатии кнопки «Смотреть полную схему общественного транспорта» выводится схема общественного транспорта Полоцка. Ссылка «Больше информации» перенаправляет пользователя на официальный сайт выбранного предприятия, если таковой имеется.

Остальные районы, представленные на сайте, имеют похожую структуру и выполняют те же функции. Переходы между районами можно осуществлять в любой момент работы сайта с помощью стрелок перехода или же перейдя ко всем районам и выбрав нужный. При нахождении и выборе объектов через поисковую форму осуществляется автоматический переход на нужный район города и вывод информации по необходимому объекту.

Разработанный программный продукт рассчитан на самую различную аудиторию, в частности туристические агентства, туроператоры, самостоятельные туристические группы, жители города, любознательные люди, желающие больше узнать о самом древнем городе Беларуси с целью упрощения процедуры поиска и составления туристических маршрутов через город Полоцк, также оказания помощи в изучении и сборе актуальной и полезной информации о главных объектах города.

Реализованный сайт является конкурентоспособным на рынке информационных технологий, так как имеет малую себестоимость, достаточный функционал и широкую информационную базу о главных туристических объектах, в отличие от существующих аналогов.

Разработанный путеводитель является легко расширяемым информационным ресурсом, что позиционирует его как проект с потенциалом развития в более крупную систему или интеграцию в уже существующую систему как самостоятельного модуля (например, как часть официального сайта города Полоцка).

Таким образом, можно сделать **вывод**, что на сегодняшний день технология WebGL, несмотря на ее недостатки, является самой простой для использования, производительной, оптимизированной и актуальной web-технологией для представления трехмерной графики в браузерах и Web-приложениях. Применение данной технологии предоставляет возможность разработки простого и удобного 3D-путеводителя, обладающего достаточной функциональностью для решения поставленных перед ним задач. Разработанный интерактивный путеводитель имеет удобную навигацию, большую базу главных объектов города, предоставляет систематизированную информацию о них, полностью адаптирован под русскоязычное население и является бесплатным программным продуктом.

ЛИТЕРАТУРА

1. WebGL – Интерфейсы веб API | MDN [Электронный ресурс]. – Режим доступа: http://www.developer.mozilla.org/ru/docs/Web/API/WebGL_API. – Дата доступа: 23.09.2017.
2. Введение в WebGL [Электронный ресурс]. – Режим доступа: <http://metanit.com/web/webgl/1.1.php>. – Дата доступа: 20.09.2017.
3. Разработка браузерной 3D-графики (графики для браузера), WebGL [Электронный ресурс]. – Режим доступа: <http://www.artofweb.ru/services/webgl-3d/>. – Дата доступа: 24.09.2017.

УДК 004:056.52

ВИДЫ АУТЕНТИФИКАЦИИ

О.И. РАЧИЦКИЙ

(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

Рассматриваются разнообразные типы и виды аутентификации, представленные в сфере безопасности веб-приложений. Отмечается наличие большого количества других протоколов и видов авторизации. Однако в данной работе речь пойдет о самых значимых, которые оказали влияние на индустрию и стали стандартами.

В целом процесс аутентификации представляет собой ответ на вопрос, «является ли клиент тем, за кого себя выдает». Проверки подлинности личности клиента осуществляются по самым различным протоколам и самыми различными способами.

В зависимости от степени доверительных отношений, структуры, особенностей сети и удаленности объекта проверка может быть односторонней или взаимной. Также различают однофакторную и строгую (криптографическую) аутентификации. Из однофакторных систем, наиболее распространенными на данный момент являются парольные системы аутентификации. У пользователя есть идентификатор и пароль, то есть секретная информация, известная только пользователю (и, возможно, системе), которая используется для прохождения аутентификации. В зависимости от реализации системы, пароль может быть одноразовым или многоразовым. В целом можно выделить следующие типы аутентификации: базовая, дайджест-аутентификация, с использованием средств протокола HTTPS, по предъявлению цифрового сертификата, с использованием куки; а также группа децентрализованных типов аутентификации: OAuth 1.0, OAuth 2.0, OpenAuth, OpenID Connect.

При использовании *базовой аутентификации* имя пользователя и пароль включаются в состав запроса. Любой злоумышленник, перехвативший пакет, способен получить доступ к секретной информации. Из ключевых недостатков такой аутентификации следует отметить невысокий уровень безопасности – пароль можно подсмотреть, угадать, подобрать, сообщить посторонним лицам, получить посредством перехвата сообщений между клиентом и аутентификационным сервером.

Далее рассмотрим *дайджест-аутентификацию*, при которой пароль пользователя передается в хешированном виде. Казалось бы, что по уровню безопасности передачи паролей этот тип мало чем отличается от базовой аутентификации, так как злоумышленнику предоставляется возможность перехватить сообщение в любом его виде, и злоумышленник все равно получает доступ к конечной точке. Но это не совсем так – пароль превращается в некоторый хеш по определенному алгоритму и всегда с добавлением произвольной строки символов, которая генерируется для каждого соединения. Таким образом, при каждом соединении генерируется новый хэш пароля и перехват его ничего не даст.

Протокол *HTTPS* позволяет шифровать все данные, передаваемые между браузером и сервером, а не только имена пользователей и пароли. Протокол HTTPS следует использовать в любом случае для обеспечения безопасности передаваемых данных. Это поддерживается практически всеми современными облачными платформами, включая выбранную для развертывания системы Heroku.

Механизмы аутентификации с применением *цифровых сертификатов*, как правило, используют протокол с запросом и ответом. Сервер аутентификации отправляет пользователю последовательность символов, так называемый запрос. В качестве ответа выступает запрос сервера аутентификации, подписанный с помощью закрытого ключа пользователя.

Множество различных сервисов и старых сайтов используют в качестве средства аутентификации *куки*. Если куки удастся похитить, то, подделав эти данные, можно аутентифицироваться в качестве другого пользователя. В случае, когда вводимые данные плохо фильтруются или не фильтруются вовсе, похитить куки становится очень просто. Для защиты куки от похищения методом запуска вредоносных JavaScript скриптов в браузере клиента используется флаг HTTPOnly, который поддерживают современные браузеры, однако это гарантирует защиту только внутри браузера, для защиты куки от перехвата во время передачи по незащищенным каналам используется флаг Secure, который шифрует куки средствами браузера. Тем не менее даже в этом случае куки все равно остаются уязвимостью с точки зрения безопасности аутентификации, так как браузер будет отправлять все куки с подходящим доменом при запросе на любой сайт из этого домена. На этой особенности построены CSRF (cross-request forgery) атаки.

Далее можно рассмотреть *децентрализованную аутентификацию*. Одним из главных минусов таких систем является то, что взлом дает доступ сразу ко многим сервисам. Однако взамен они предоставляют наивысший уровень безопасности, так как они изначально созданы и нацелены на осуществление

наибезопаснейшей аутентификации пользователей с созданием необходимых сессий и использованием самых последних разработок в области шифрования и безопасности.

OpenID – это открытая децентрализованная система аутентификации пользователей, она позволяет пользователю иметь одни аутентификационные данные для различных сервисов. Безопасность обеспечивается подписью всех сообщений. Возможной уязвимостью OpenID является подверженность атакам посредством перехвата запросов.

Многофакторная аутентификация используется для повышения безопасности посредством использования нескольких факторов аутентификации сразу. Но не всякая комбинация нескольких методов является многофакторной аутентификацией. Используются факторы различных типов: отличительные свойства (сканирование лица, отпечаток пальцев, радужная оболочка глаз); некоторое знание, которым обладает клиент (пароль, пин-код, последовательность действий); физическая вещь (карта доступа, пропуск, флеш-память, сим-карта с определенным телефонным номером).

Использование классических многоразовых паролей является серьезной уязвимостью при использовании открытых систем или незащищенных протоколов передачи данных. Это подтолкнуло разработчиков систем безопасности к созданию генераторов одноразовых паролей. Такие устройства или программы генерируют пароли для каждого соединения или сессии, чтобы при разрыве или перехвате сессии было невозможно подделать личность клиента. Для сохранения принципа двухфакторности аутентификации помимо сгенерированного устройством значения пользователь вводит постоянный пароль или подтверждает свой вход в систему вводом кода SMS-сообщения, которое на его личный номер высылает система.

Заключение

В результате проведенного исследования рассмотрены основные виды проверки подлинности клиентов на стороне приложений. Показано, что существует также много других видов аутентификации, более или менее популярных, каждый со своими плюсами и минусами. Очевидно, что правильное комбинирование систем аутентификации может значительно усложнить кражу данных клиентов.

ЛИТЕРАТУРА

1. Authentication Cheat Sheet [Электронный ресурс]. – Режим доступа: https://www.owasp.org/index.php/Authentication_Cheat_Sheet. – Дата обращения: 26.09.2017.
2. Understanding Authentication, Authorization, and Encryption [Электронный ресурс]. – Режим доступа: <https://www.bu.edu/tech/about/security-resources/bestpractice/auth/>. – Дата обращения: 26.09.2017.
3. Understanding and selecting authentication methods [Электронный ресурс]. – Режим доступа <http://www.techrepublic.com/article/understanding-and-selecting-authentication-methods/>. – Дата обращения: 26.09.2017.
4. Аутентификация и авторизация в микросервисных приложениях [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/company/dataart/blog/311376/>. – Дата обращения: 26.09.2017.

УДК 004:056.53

УЯЗВИМОСТИ В СОВРЕМЕННОМ КЛИЕНТ-СЕРВЕРНОМ ОБЩЕНИИ**О.И. РАЧИЦКИЙ***(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)*

Описаны основные виды атак на веб-приложения, которые используют уязвимости протоколов сообщения между клиентами и серверами, или недочеты архитектуры некоторых приложений, дизайн которых был создан без учета уязвимостей.

Современные браузеры являются достаточно защищенными от практически любых действий злоумышленников, ставящих своей целью похитить данные пользователей или воспользоваться пользовательскими полномочиями. Но в силу особенностей клиент-серверного взаимодействия структуры запросов между клиентом и сервером и некоторых особенностей JavaScript, используемого как язык работы с сайтом с клиентской стороны, существуют типы атак, которые невозможно предотвратить только силами браузера. О них и пойдет речь в этой работе.

Наиболее популярная и простая атака – это CSRF (cross-request forgery) атака. При CSRF атаке злоумышленники заставляют клиента выполнить нежелательные действия в приложениях, в которых клиент аутентифицирован. Этот тип атаки основан на недостатках протокола HTTP. Алгоритм действия злоумышленников в данном случае простой:

- найти на целевом сайте GET или POST запрос, выполнение которого клиентом может вызвать интерес со стороны злоумышленников. Это может быть запрос на денежный перевод или передачу личных данных;
- сформировать запрос таким образом, чтобы целевой сайт отослал данные или осуществил денежный перевод в том виде, в котором это выгодно мошенникам;
- создать собственный вредоносный сайт и поместить туда скрипт, который автоматически будет или осуществлять перенаправление на нужный сайт (для GET запросов), или активировать отправку формы с нужными параметрами (для POST запросов);
- дать ссылку в каком-либо виде пользователю или заставить пользователя перейти по ссылке, используя вредоносный код или скрипт, встроенный каким-либо образом в браузер клиента;
- как только пользователь перейдет на вредоносный сайт, на этот сайт не будут отправлены его куки, созданные на целевом сайте, однако как только вредоносный скрипт осуществит запрос к целевому сайту от лица пользователя, в таком запросе браузер автоматически передаст все куки домена целевого браузера, и если целью злоумышленников было проведение денежной транзакции, то целевой сайт проверит достоверность куки пользователя, а после авторизует и обработает сформированный злоумышленниками запрос. Это является основной уязвимостью при хранении аутентификационной и авторизационной информации в куки.

Вторым популярным типом атаки является XSS (Cross-Site Scripting). Суть XSS заключается в том, что разнообразными способами на страницу сайта, отображаемую у клиента, внедряется вредоносный код. Причем не всегда для этого требуется взлом самого сервера. XSS атаки можно разбить на два подтипа: отраженные и хранимые.

Отраженные XSS атаки основаны на отраженной уязвимости, на сегодняшний день является самой распространенной XSS-атакой. Эти уязвимости появляются, когда данные, предоставленные веб-клиентом, чаще всего в параметрах HTTP-запроса или в форме HTML исполняются непосредственно серверными скриптами для синтаксического анализа и отображения страницы результатов для этого клиента, без надлежащей обработки. Отраженная XSS-атака срабатывает, когда пользователь переходит по специально подготовленной ссылке, в которой в параметрах запроса передается вредоносный JavaScript код, не кодирующийся сервером и может быть исполнен в браузере клиента на странице, которую ему вернет сам сервер.

Отраженные атаки, как правило, рассылаются по электронной почте или размещаются на Web-странице. URL приманки не вызывает подозрения, указывая на надежный сайт, но содержит вектор XSS. Если доверенный сайт уязвим к вектору XSS, то переход по ссылке может привести к тому, что браузер жертвы начнет выполнять встроенный скрипт.

Хранимые XSS атаки являются наиболее разрушительными, потому что они возможны только тогда, когда злоумышленнику удастся внедрить на сервер вредоносный код, выполняющийся в браузере каждый раз при обращении к оригинальной странице. Классическим примером этой уязвимости являют-

ся форумы, блоги, чаты, на которых разрешено оставлять комментарии в HTML формате без ограничений. Также эта уязвимость имеет место быть, когда на сервере разрешается хранить пользовательские тексты и рисунки. Скрипты вставляются в эти тексты и рисунки.

Еще одной распространенной ранее, но возможной даже в современном мире веб-технологий, является *SQL-инъекция*. Внедрение SQL в зависимости от типа используемой СУБД и условий внедрения может дать возможность атакующему выполнить произвольный запрос к базе данных (например, прочитать содержимое любых таблиц, удалить, изменить или добавить данные), получить возможность чтения и/или записи локальных файлов и выполнения произвольных команд на атакуемом сервере.

Некоторые сервера опираются на клиентскую часть в плане составления запросов и не проверяют передаваемые параметры. Это может быть уязвимостью при внедрении в параметры SQL-кода, который будет исполнен сервером.

Также даже при защите от банального внедрения SQL-кода разработчики часто забывают о наличии оператора UNION в большинстве SQL СУБД, который призывает сервер выполнить помимо основной команды еще и дополнительную, описанную после этого оператора. Если говорить другими словами, язык SQL позволяет объединять результаты нескольких запросов при помощи оператора UNION. Это предоставляет злоумышленнику возможность получить несанкционированный доступ к данным.

Зачастую SQL-запрос, подверженный данной уязвимости, имеет структуру, усложняющую или препятствующую использованию UNION. В таких случаях злоумышленниками используется метод экранирования части запроса при помощи символов комментария в зависимости от типа СУБД. Для разделения команд в языке SQL используется символ *точка с запятой*, внедряя этот символ в запрос, злоумышленник получает возможность выполнить несколько команд в одном запросе, однако не все диалекты SQL поддерживают такую возможность.

Поиск скриптов, уязвимых для атаки, осуществляется злоумышленниками посредством изучения поведения скриптов сервера при манипуляции входными параметрами с целью обнаружения их аномального поведения. Манипуляция происходит всеми возможными параметрами: данными, передаваемыми через методы POST и GET, значениями куки, HTTP_REFERER (для скриптов), AUTH_USER и AUTH_PASSWORD (при использовании аутентификации). Как правило, манипуляция сводится к подстановке в параметры символа одинарной (реже двойной или обратной) кавычки.

Аномальным поведением считается любое поведение, при котором страницы, получаемые до и после подстановки кавычек, различаются (и при этом не выведена страница о неверном формате параметров). Наиболее частые примеры аномального поведения – выводится сообщение о различных ошибках, при запросе данных запрашиваемые данные не выводятся вообще, хотя страница отображается. Следует учитывать, что известны случаи, когда сообщения об ошибках в силу специфики разметки страницы не видны в браузере, хотя и присутствуют в ее HTML-коде.

Заключение

Несмотря на известность атак и общую осведомленность об их существовании избавиться от них невозможно из-за особенностей стандартов HTTP и модели серверов. Единственным вариантом защиты от такого типа атак является их предвидение и контролирование необходимых для проведения атак условий на стороне сервера. Об этом следует помнить разработчикам сайтов.

ЛИТЕРАТУРА

1. Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), SQL Injection, HTML Injection, Etc [Электронный ресурс]. – Режим доступа: <https://www.zscaler.com/blogs/research/cross-site-scripting-xss-cross-site-request-forgery-csrf-sql-injection-html-injection-etc> . – Дата доступа: 26.09.2017.
2. CSRF and XSS: A Lethal Combination – Part I [Электронный ресурс]. – Режим доступа: <http://resources.infosecinstitute.com/csrf-xss-lethal-combination/>. – Дата доступа: 26.09.2017.
3. Understanding and selecting authentication methods [Электронный ресурс]. – Режим доступа: <http://www.techrepublic.com/article/understanding-and-selecting-authentication-methods/>. – Дата доступа: 26.09.2017.
4. XSS глазами злоумышленника [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/post/66057>. – Дата доступа: 26.09.2017.

УДК 004

**ЦЕЛЕСООБРАЗНОСТЬ ИСПОЛЬЗОВАНИЯ ТЕХНОЛОГИИ ASP.NET
И ЕЕ ПЛАТФОРМЫ WEBAPI ПРИ РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЙ**

С.Н. МАЛАШОНОК,
(Представлено: Ю.Н. КРАВЧЕНКО)

Рассматривается технология разработки ASP.NET, которая является частью платформы NetFramework и предназначена для написания клиент-серверных приложений и веб-сервисов, и платформы WebAPI, которая является частью ASP.NET. Анализируются целесообразность и преимущества ее использования.

ASP.NET представляет собой технологию, направленную на создание серверных веб-приложений и веб-сервисов различной сложности. Она содержит большое количество готовых элементов управления и различных технологий, начиная от технологии, предоставляющий доступ к базам данных и заканчивая технологией создания распределенных приложений.

Технология развивалась из более ранней версии технологии MicrosoftASP. Технология ASP имела ограничения по своим возможностям. Код программы вставлялся прямо в HTML-страницу в виде специальных тегов, что было неудобно и создавало путаницу.

Пример такой HTML-страницы представлен на рисунке 1.

```
<%@ Language=VBScript %>
<% Option Explicit%>
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" content="text/html">
<TITLE>
Hello ASP World!
<TITLE>
</HEAD>
<BODY>
<P><%
Dim i;
for i=1 to 5
Response.Write("<FONT SIZE=" & i)
Response.Write(">Этот код генерирует ASP!</FONT>")
next i
%></P>
</BODY>
</HTML>
```

Рисунок 1. – Пример файла ASP

Microsoft полностью перестроила ASP.NET, основываясь на CommonLanguageRuntime (CLR), которая является основой всех приложений Microsoft .NET. Разработчики могут писать код для ASP.NET, используя практически любые языки программирования, входящие в комплект .NET Framework (C#, Visual Basic.NET и JScript.NET) [1]. Как и все приложения .NET, приложения ASP.NET всегда компилируются. На первом этапе компиляции код компилируется в промежуточный язык IL, который на втором этапе компилируется в код на низкоуровневом машинном языке. IL-код создается только один раз и генерируется снова только при изменении исходного кода, а файлы, содержащие машинный код, кэшируются в системном каталоге. Благодаря этому механизму ASP.NET приложения не нуждаются в выполнении компиляции при каждом запросе веб-страницы.

В отличие от ASP, ASP.NET является объектно-ориентированной технологией, которая не только предоставляет коду доступ ко всем объектам .NETFramework, но и позволяет использовать все концепции объектно-ориентированного программирования. Технология позволяет создавать повторно используемые классы, использовать интерфейсы, расширять классы с помощью механизма наследования.

ASP.NET предоставляет возможность кэшировать ответы, сгенерированные страницей ASP.NET. Кэширование позволяет обработать запросы к одной и той же странице с помощью кэша, при этом код, с помощью которого была создана страница, не будет выполняться повторно.

В ASP.NET3.5 появилась технология LanguageIntegratedQuery(LINQ), которая представляет собой набор расширений для языков C# и VisualBasic. Она позволяет писать на C# или VisualBasic код, способный манипулировать находящимися в памяти данными во многом в той же манере, что и запросы к базе данных. Самой главной разновидностью LINQ является LINQ to Objects, которая позволяет брать коллекцию объектов и выполнять запрос, извлекающий определенную информацию из некоторых этих объектов [2].

С выходом ASP.NET4.5 появился новый инструмент для создания веб-служб – WebAPI, который позволяет создавать HTTP-службы, доступные для широкого круга клиентов, включая браузеры и мобильные устройства.

Упрощение создания веб-служб является неотъемлемой особенностью Web API. Оно представляет собой значительное улучшение по сравнению с другими технологиями построения веб-служб, которые компания Microsoft предлагала на протяжении последнего десятилетия [3].

С помощью WebApi можно создать библиотеку классов, которая будет обрабатывать HTTP-запросы. Данная библиотека может быть расположена в исполняющей среде, благодаря чему к ней можно будет обращаться по HTTP. Открытые методы контроллеров являются конечными точками HTTP, которые могут возвращать данные в виде двоичных данных, JavaScriptObjectNotation (JSON), eXtensibleMarkupLanguage (XML) или обычного текста. Регулируемые правила маршрутизации позволяют задать форму UniformResourceLocator (URL), что значительно упрощает передачу данных в методы контроллера. Пример класса контроллера с методом представлен на рисунке 2.

```
public class ValuesController : ApiController
{
    // GET api/values
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }

    // GET api/values/5
    public string Get(int id)
    {
        return "value";
    }
}
```

Рисунок 2. – Метод контроллера

Таким образом, к основным преимуществам ASP.NET относятся:

- компиляция кода;
- использование объектно-ориентированного программирования;
- кэширование страниц;
- большое количество стандартных объектов, предоставляемых платформой .NetFramework;
- удобное создание веб-сервисов.

ЛИТЕРАТУРА

1. Принципы ASP.NET [Электронный ресурс]. – Режим доступа: https://studopedia.ru/2_30490_printsipi-ASPNET.html. – Дата доступа: 16.09.2017.
2. Мак-Дональд, М. MicrosoftASP.NET 4 с примерами на C# 2010 для профессионалов / Мэтью Мак-Дональд, Адам Фримен, Марио Шпуста. – 4-е изд. ; под ред. Ю.Н. Артеменко. – М. : И.Д. Вильямс, 2010. – 1424 с.
3. Одностраничные приложения [Электронный ресурс]. – Режим доступа https://professorweb.ru/my/ASP.NET/mvc/level8/8_1.php. – Дата доступа: 21.09.2017.

УДК 004.514

**РАЗРАБОТКА ИНТЕРФЕЙСА WEB-ПРИЛОЖЕНИЯ ДЛЯ ОПТИМИЗАЦИИ
ФОРМИРОВАНИЯ ГРАФИКОВ ОТПУСКОВ СОТРУДНИКОВ КОМПАНИИ****С.Н. МАЛАШОНОК,**
(Представлено: Ю.Н. КРАВЧЕНКО)

Рассматриваются способ создания пользовательского интерфейса web-приложения для оптимизации формирования графиков отпусков сотрудников компании. Интерфейс реализован с применением AngularJsFramework.

Управление отпусками работников – неотъемлемая часть менеджмента на предприятии. Наиболее ярко это выражается, когда предприятие работает с множеством проектов, над которыми трудится отдельная группа работников. Чтобы достичь максимальной производительности, необходимо четко следить за состоянием группы, так как отсутствие членов команды может негативно повлиять на весь рабочий процесс. Задача директора – составить и оптимизировать график отпусков работников, чтобы не понизить производительность работы предприятия.

Поскольку web-приложение предоставляет только часть функционала по управлению персоналом, то оно не требует большого количества web-страниц, поэтому было реализовано в виде SinglePageApplication (SPA).

SPA – это веб-приложение, размещенное на одной веб-странице, которая для обеспечения работы загружает весь необходимый код вместе с загрузкой самой страницы. Приложение такого типа появились сравнительно недавно, с началом эры HyperTextMarkupLanguage (HTML5) и SPA является типичным представителем приложений на HTML5.

HTML5 – это нечто иное, как HTML + CascadingStyleSheets (CSS3) + JavaScript и т.д. Таким образом, SPA – это приложения, написанные на языке JavaScript.

Первым плюсом стоит отметить тот факт, что приложения на SPA отлично работают на устройствах как стационарных, так и мобильных. «Большие» компьютеры, планшеты, смартфоны могут беспрепятственно работать с сайтами, построенных по принципу SPA.

Второй плюс – богатый пользовательский интерфейс. Так как веб-страница одна, построить богатый, насыщенный пользовательский интерфейс гораздо проще. Проще хранить информацию о сеансе, управлять состояниями представлений и управлять анимацией [1].

Для реализации пользовательского интерфейса использовался AngularJsFramework, который предназначен для разработки одностраничных приложений и использует шаблон Model-View-Controller (MVC).

Для описания интерфейса используется декларативное программирование, а бизнес-логика отделена от кода интерфейса, что позволяет улучшить тестируемость и расширяемость приложений.

Другой отличительной чертой фреймворка является двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом. Таким образом, AngularJS синхронизирует модель и представление.

Кроме того, AngularJs поддерживает такие функциональности, как AsynchronousJavascriptandXML (Ajax), управление структурой DocumentObjectModel(DOM), анимация, шаблоны, маршрутизация и т.д. Мощь фреймворка, наличие богатого функционала во многом повлияла на то, что он находит свое применение во все большем количестве веб-приложений [2].

На главной странице web-приложения отображается таблица с сотрудниками предприятия и количеством отработанных часов.

В верхней части страницы находятся фильтры календаря.

Поле ввода служит для фильтрации таблицы по проектам, фамилии и имени сотрудника.

Флаги используются для фильтрации календаря по типам отпусков.

Также можно построить график отпусков за любой период времени. Дни, на которые у сотрудника взят отпуск, отображаются разными цветами. По цвету можно будет определить статус заявки:

- красный – заявка не рассмотрена;
- зеленый – заявка подтверждена.

Если заявка сотрудника на отпуск накладывается на другую заявку или подтвержденный отпуск других сотрудников, работающих на совместных проектах, то заявка будет выделена специальным цветом и значком. При нажатии на значок график будет отсортирован по проектам сотрудника, который подал заявку на отпуск, и пересекающимся заявкам.

Страница графиков отпусков по месяцам представлена на рисунке 1.

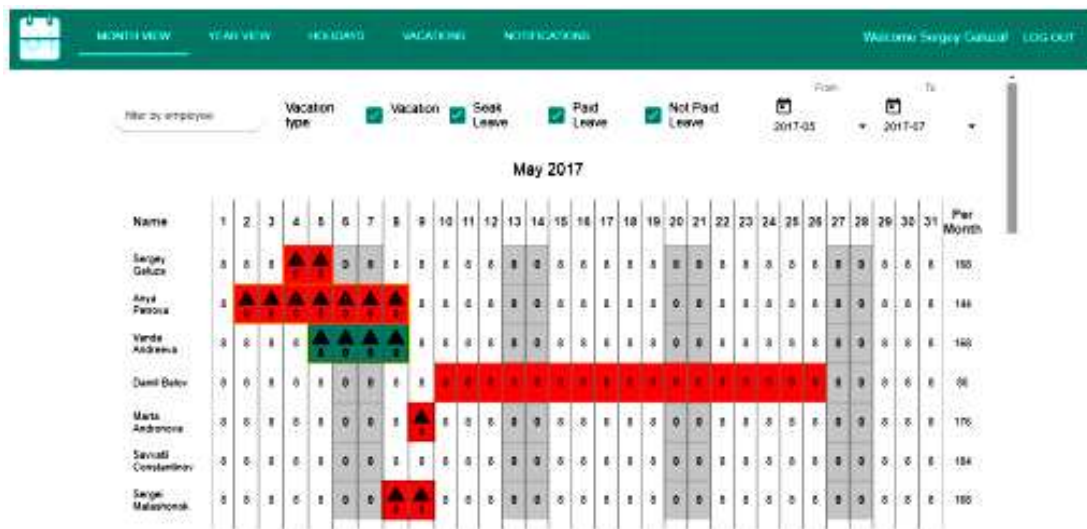


Рисунок 1. – Страница графиков отпусков по месяцам

Для учета и контроля личных отпусков разработана страница, представленная на рисунке 2.

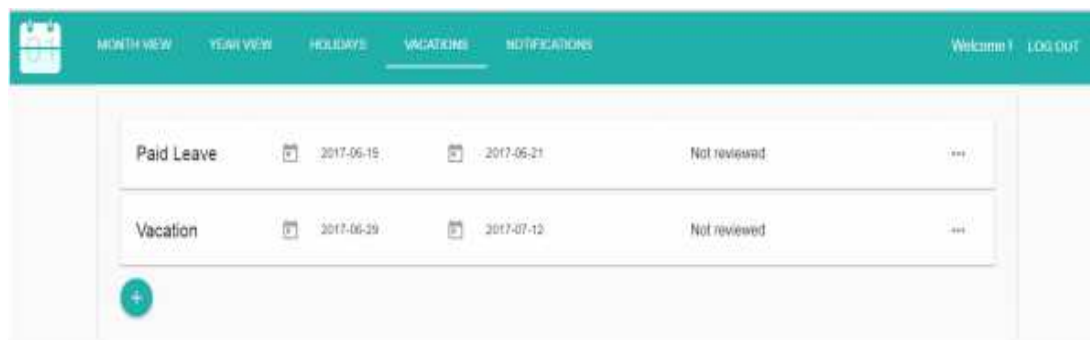


Рисунок 2. – Страница просмотра заявок на различные виды отпусков

В блоке отпуска отображаются тип отпуска, дата начала и конца, статус. Также на этой странице предусмотрена возможность создавать заявки на отпуска, после чего администратору будет выслано уведомление о том, что была создана новая заявка, а статус отпуска установится как не просмотренный.

В разработанном web-приложении реализованы функции добавления, редактирования и удаления заявок и отпусков работников предприятия, управления графиками отпусков, установки сокращенных и праздничных дней, установки количества отработанных часов сотрудниками, переноса рабочих дней, а также реализован полноценный сервис уведомлений.

Данное web-приложение должно упростить процессы формирования графиков отпусков и подачи сотрудниками заявок на отпуска.

ЛИТЕРАТУРА

1. Что такое SPA или одностраничный портал [Электронный ресурс]. – Режим доступа: <http://www.codenet.ru/webmast/js/spa/>. – Дата доступа: 18.09.2017.
2. Введение в AngularJS [Электронный ресурс] – Режим доступа: <https://metanit.com/web/angular/1.1.php>. – Дата доступа: 20.09.2017.

УДК 004.056.55

**ПРОЕКТИРОВАНИЕ СТЕГОСИСТЕМЫ,
ОСНОВАННОЙ НА СОКРЫТИИ ТЕКСТОВЫХ ДАННЫХ
В АУДИОФАЙЛАХ ЗА СЧЕТ ИЗМЕНЕНИЯ ВРЕМЕНИ ЗАДЕРЖКИ ЭХО-СИГНАЛА**

А.В. КОХАНОВСКИЙ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматривается практический способ создания надежной стегосистемы с нестандартным способом скрытия информации. Построена система, основанная на скрытии текстовых сообщений в аудиофайлах. Выяснено, насколько пригодна такая система для практического применения.

В статье пойдет речь о сокрытии данных в аудиофайлах с помощью одного из методов стеганографии. Под цифровой стеганографией понимается скрытие одной информации в другой. Причем сокрытие это должно реализовываться таким образом, чтобы, во-первых, не были утрачены свойства и некоторая ценность скрываемой информации, а во-вторых, неизбежная модификация информационного носителя, не только не уничтожила смысловые функции, но и на определенном уровне абстракции даже не меняла их. Тем самым факт передачи одного сообщения внутри другого не выявляется традиционными методами.

Принцип построения стеганографической системы и описание алгоритма скрытия данных. Стандартная стеганографическая схема сохраняется вне зависимости от технологии, которой она реализуется. Задачу встраивания и выделения сообщений из другой информации выполняет стегосистема. Стегосистема состоит из следующих основных элементов, представленных на рисунке 1.

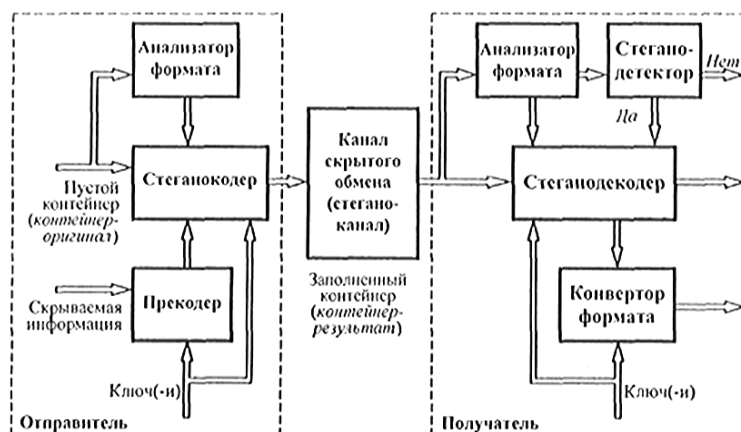


Рисунок 1. – Структурная схема типичной стегосистемы

Контейнер – любая информация, предназначенная для сокрытия тайных сообщений. Стегоключ, или просто ключ, – секретный ключ, необходимый для сокрытия информации. *Прекодер* – устройство, предназначенное для преобразования скрываемого сообщения к виду, удобному для встраивания в сигнал-контейнер. *Стегокодер* – устройство, предназначенное для осуществления вложения скрытого сообщения в другие данные с учетом их модели; устройство выделения встроенного сообщения. *Стегодетектор* – устройство, предназначенное для определения наличия стегосообщения. *Декодер* – устройство, восстанавливающее скрытое сообщение. Этот узел может отсутствовать.

Данные, содержащие скрытое сообщение, могут подвергаться преднамеренным атакам или случайным помехам. Как показано на рисунке 1, в стегосистеме происходит объединение двух типов информации так, чтобы они могли быть различимы двумя принципиально разными детекторами. В качестве одного из детекторов выступает система выделения цифровых водяных знаков (ЦВЗ), в качестве другого – человек [1].

Для сокрытия данных использовался метод встраивания информации за счет изменения времени задержки эхо-сигнала. Этот метод позволяет внедрять данные в сигнал прикрытия, изменяя параметры эхо-сигнала. К параметрам эхо, несущим внедряемую информацию, относятся: начальная амплитуда, время спада и сдвиг (время задержки между исходным сигналом и его эхо). При уменьшении сдвига два сигнала смешиваются. В определенной точке человеческое ухо перестает различать два сигнала, и эхо воспринимается, как добавочный резонанс. Кодер использует два времени задержки: одно для кодирования нуля, другое для кодирования единицы. И то, и другое время задержки меньше того, на котором человеческое ухо может распознать эхо. Кроме уменьшения времени задержки необходимо добиться уста-

новлением начальной амплитуды и времени спада того, чтобы внедренная информация не могла быть воспринята системой слуха человека.

При кодировании информации были выбраны значения задержек эхо-сигналов, равные 0,0012 и 0,0008 секунды. Данные значения снижают эффективность алгоритма извлечения, уменьшая вероятность правильного извлечения бита, однако подходит практически для всех типов контейнеров. Значение амплитуды, равное 40%, выбрано в связи с тем, что оно помогает увеличить вероятность того, что факт наличия скрытой информации не будет раскрыт.

Декодирование внедренной информации представляет собой определение промежутка времени между сигналом и эхо. Для этого необходимо рассмотреть амплитуду (в двух точках) автокорреляционной функции дискретного косинусного преобразования логарифма спектра мощности (кепстра). Всплеск автокорреляционной функции будет наблюдаться через δ_1 или δ_0 секунд после исходного сигнала (рис. 2). Правило декодирования основано на определении промежутка времени между исходным сигналом и всплеском автокорреляции [2].

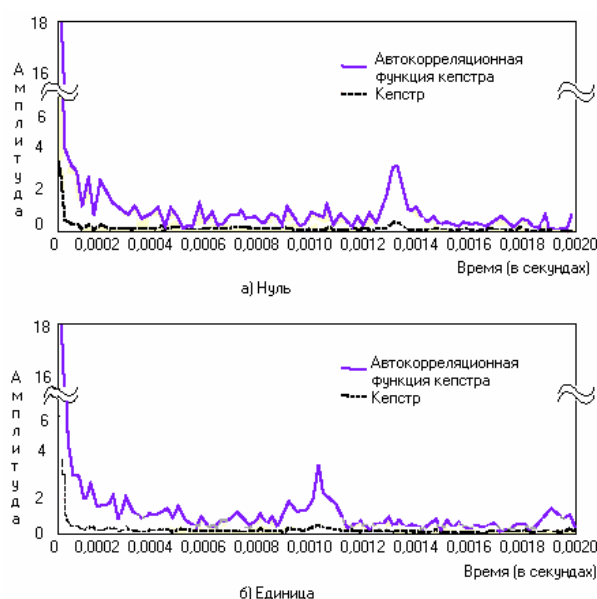


Рисунок 2. – Поведение автокорреляционной функции

По исследованиям В. Бендера и Н. Моримото, данная схема позволяет внедрять 16 бит в одну секунду аудиозаписи незаметно, без потери ее качества [3]. Выбор метода встраивания информации в звуковые файлы, используя преобразования эхо-сигналов, обусловлен тем, что этот метод имеет устойчивость к амплитудным и частотным атакам, что позволяет обойти остальные, неустойчивые к этим атакам, методы. Однако данный метод неустойчив к временным атакам, напрямую влияющим на длину аудио-файла и количество отсчетов в нем. При реализации метода стеганографической защиты информации посредством преобразования эхо-сигналов стало ясно, что основную трудность представляет реализация наиболее эффективного алгоритма извлечения встроенных бит. Некоторые исследования в этом направлении показали, что возможно достичь наибольшей эффективности только посредством индивидуального подхода к каждому контейнеру отдельно, меняя при этом время задержек накладываемого эхо-сигнала.

Заключение. Исследования в области стеганографии – перспективное направление защиты информации, так как в современном мире задача передачи секретной информации стоит наравне со скрытым общением, т.е. скрытия факта передачи сообщений. Поэтому необходимо продолжать исследованиями в этой области для поиска новых, эффективных, методов или улучшения существующих. В данной работе был рассмотрен метод встраивания информации в звуковые файлы. Метод эхо-сигналов – самый перспективный, однако требует доработки в плане пропускной способности, и вероятности правильного извлечения встроенных бит информации. Программный продукт реализован и готов к использованию с возможностью доработки.

ЛИТЕРАТУРА

1. Matsui. K. Digital signature on a facsimile document by recursive MH coding / K. Matsui, K. Tanaka and Y. Nakamura // Symposium On Cryptography and Information Security, 1989.
2. Садов, В.С. Компьютерная стеганография / В.С. Садов. – М. : МГВРК, 2012. – 289 с.
3. Грибунин, В.Г. Цифровая стеганография / В.Г. Грибунин. – М. : СОЛОН-Пресс, 2002. – 272 с.

УДК 004.056.55

**ОПРЕДЕЛЕНИЕ СТЕПЕНИ ПРИГОДНОСТИ КОНТЕЙНЕРА
ДЛЯ СТЕГАНОГРАФИЧЕСКОЙ МОДИФИКАЦИИ В СТЕГОСИСТЕМЕ,
ОСНОВАННОЙ НА СОКРЫТИИ ТЕКСТОВЫХ ДАННЫХ В АУДИОФАЙЛАХ
ЗА СЧЕТ ИЗМЕНЕНИЯ ВРЕМЕНИ ЗАДЕРЖКИ ЭХО-СИГНАЛА**

А.В. КОХАНОВСКИЙ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Определяется степень пригодности контейнера для стеганографической модификации. Представлены некоторые виды атак на стегосистему. Построена система, основанная на скрытии текстовых сообщений в аудиофайлах, а также изучены атаки и выяснено, насколько пригодна такая система для практического применения.

Основными параметрами аудиосигналов являются их амплитуда, частота и фаза. Для стеганографической модификации пригодны все эти параметры, но перед процедурой встраивания необходимо оценить пределы модификации параметров того или иного контейнера, а также вносимые при этом искажения. Пропускная способность, надежность, устойчивость стеганографической системы во многом будет определяться степенью модификации аудиоконтейнера [1].

Оценка стеганостойкости реализованной системы

Существует множество классификаций оцифрованных аудиосигналов по их техническим параметрам, таким как, частота дискретизации, количество каналов (моно или стерео), качество записи (количество кб в секунду) и т.д. В то же время, если аудиосигналы (контейнеры) принадлежат к одному формату представления данных (например, WAVE) имеют одну частоту дискретизации, одинаковое количество каналов, схожее качество звука и различаются только по музыкальному жанру, достаточно сложно оценить пригодность того или иного контейнера к стеганографической модификации.

Все аудиосигналы можно разбить на три группы (рис. 1):

- аудиофайлы, подобные речи, с большим количеством пауз: стихи, голос диктора, речь с негромкой фоновой музыкой и т.д. (рис. 1, а);
- аудиофайлы, не обладающие широким частотным диапазоном, подобные обычным музыкальным композициям: эстрадные песни, популярная музыка, блюз и т.д. (рис. 1, б);
- аудиофайлы, обладающие широким частотным диапазоном, прошедшие дополнительную компьютерную обработку: электронная музыка (trance, house, techno и др.), синтезированные звуки природы и т.д. (рис. 1, в).

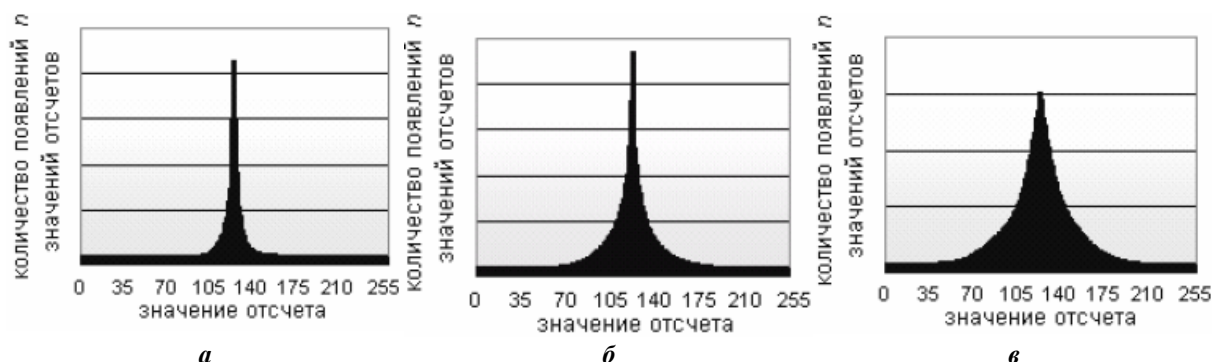


Рисунок 1. – Графики распределения отсчетов для трех групп аудиосигналов

Использование аудиофайлов из первой группы оказалось совсем неприемлемым для использования независимо от качества использованного контейнера. Вероятность правильного и полного изъятия сообщения из контейнера достаточно мала. При использовании аудиофайлов из второй группы, как и ожидалось, искажения едва различимы натренированным ухом, а вероятность правильного изъятия достаточно велика. Использование аудиофайлов из третьей группы показало, что вероятность правильного извлечения близка к 100%, при этом различить пустой контейнер и контейнер с данными на слух не удастся.

Из этого можно сделать вывод, что лучше всего для скрытия данных методом задержки эхо-сигнала подходят аудиофайлы, прошедшие дополнительную компьютерную обработку (электронная му-

зыка). Также следует учесть частоту дискретизации и количество каналов: чем их значения выше, тем больше вероятность правильного изъятия [2].

Основные виды атак на стегосистемы:

1. *Атака на основе известного заполненного контейнера.* Задача состоит в обнаружении факта наличия скрытого сообщения. Прослушивание аудиосигнала с целью обнаружить какие-либо помехи. Был выбран контейнер длительностью 60 секунд, с динамичной музыкой. И произведено скрытие сообщения длиной 20 символов. Результат – прослушивание заполненного контейнера на акустической системе показало, что никаких посторонних звуков, хрипов и тресков не обнаружено.

2. *Атака на основе известного пустого контейнера.* Если он известен, то путем сравнения его с предполагаемым контейнером, в котором есть скрытое сообщение, всегда можно установить факт наличия стегоканала. Это может быть сравнение по длине аудиосигнала, сравнение по размеру, а также прослушивание. Был выбран контейнер длительностью 60 секунд, с динамичной музыкой. И произведено скрытие сообщения длиной 20 символов. Результат – многократное прослушивание заполненного контейнера и контейнера оригинала на акустической системе без проигрывания низких частот (ноутбук) показало, что отличий между звучанием нет. Однако при прослушивании на акустической системе с устройством, проигрываемым низкие частоты, стали заметны некоторые отличия в глубине звучания заполненного контейнера и контейнера оригинала.

3. *Атаки против встроенного сообщения,* направленные на удаление или порчу встроенной информации путем манипулирования заполненным контейнером, а также направленные на затруднение или невозможность правильной работы детектора. При этом сообщение в аудиофайле остается, но теряется возможность его приема. Примерами являются: обрезка (изменение длины) аудиосигнала; изменение формата на другой; изменение параметров формата контейнера, ускорение или замедление; изменение тональности. Результат – метод скрытия данных с помощью задержки эхо-сигнала неустойчив к временным изменениям. При изменении длины аудиофайла на несколько секунд извлечь какую-либо информацию уже невозможно. Изменение любых других параметров аудиофайла не повлияло на получение скрытой информации [3].

Заключение

Рассмотрен метод встраивания информации в звуковые файлы. Метод эхо-сигналов – самый перспективный, однако требует доработки в плане пропускной способности и вероятности правильного извлечения встроенных бит информации. Для оценки эффективности рассматривались такие параметры, как особые требования к контейнерам – звуковым файлам и сложность определения ее злоумышленником, влияние попытки уничтожения скрытой информации на сохранность контейнера. Выбор метода встраивания информации в звуковые файлы, используя преобразования эхо-сигналов, обусловлен тем, что этот метод имеет устойчивость к амплитудным и частотным атакам, что позволяет обойти остальные, неустойчивые к этим атакам, методы. Однако данный метод неустойчив к временным атакам, напрямую влияющим на длину аудиофайла и количество отсчетов в нем.

Программа реализована и готова к использованию с возможностью ее доработки.

ЛИТЕРАТУРА

1. Садов, В.С. Компьютерная стеганография / В.С. Садов. – М. : МГВПК, 2012. – 289 с.
2. Matsui, K. Digital signature on a facsimile document by recursive MH coding / K. Matsui, K. Tanaka and Y. Nakamura // Symposium On Cryptography and Information Security, 1989.
3. Грибунин, В.Г. Цифровая стеганография / В.Г. Грибунин. – М. : СОЛОН-Пресс, 2002. – 272 с.

УДК 004.021

LibGDX – КРОССПЛАТФОРМЕННЫЙ ФРЕЙМВОРК**Р.А. КАРЦЕВИЧ***(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматривается кроссплатформенный фреймворк, применяемый в создании различного рода приложений. Показано, что кроссплатформенный фреймворк LibGDX является отличным инструментом в создании игровых приложений благодаря своей простоте и использованию таких языков, как C и C++, для обеспечения более быстрой работы.

В основном создание игровых приложений подразумевает за собой использование какого-либо фреймворка. *Фреймворки* – это программные продукты, которые упрощают создание и поддержку технически сложных или нагруженных проектов. Фреймворк, как правило, содержит только базовые программные модули, а все специфичные для проекта компоненты реализуются разработчиком на их основе.

Плюсы фреймворков:

- разработка на фреймворке (в отличие от самописных решений) позволяет добиться простоты сопровождения проекта;
- возможна (и относительно проста) реализация любых бизнес-процессов, а не только тех, которые изначально заложены в систему. Также проекты на базе фреймворков легко масштабируемы и модернизируемы;
- решения на фреймворках, как правило, работают значительно быстрее и выдерживают большую нагрузку, чем CMS и самописные системы. Именно поэтому много популярных интернет-магазинов работают не на коробочных CMS, а на фреймворках. По уровню безопасности решения на фреймворках значительно превосходят самописные системы и сравнимы с CMS (как правило, сайты на фреймворках даже безопаснее).

Минусы фреймворков:

- сроки разработки типового функционала на фреймворках больше, чем при использовании CMS. Фреймворки содержат только базовые компоненты бизнес-логики уровня приложения, поэтому многие функции реализовываются индивидуально;
- для разработки на фреймворке требуется понимание бизнес-процессов, которые требуется реализовать. Например, если в CMS уже есть некий предустановленный процесс обработки заказов, то фреймворки такого не предоставляют;
- одним из достойных кандидатов является фреймворк LibGDX, благодаря которому облегчается создание кроссплатформенных приложений.

LibGDX – кроссплатформенный фреймворк для разработки игр и визуализации, основанный на языке программирования Java с некоторыми компонентами, написанными на C и C++ для повышения производительности определенного кода. В настоящее время поддерживает Windows, Linux, Mac OS X, Android, iOS и HTML5 как целевые платформы.

Одними из выдающихся возможностей фреймворка являются:

- поддержка Windows, Linux, Mac OS X, Android ОС, Браузеры с поддержкой WebGL и iOS.32 и 64-разрядные версии ОС;
- гибкость и увеличение возможностей достигается за счет расширений. Можно подключить физический движок Box2D для работы с объектами и физикой реального мира, добавить поддержку TrueType шрифтов или работать с 3D-объектами;
- встроенный набор виджетов для рисования графического интерфейса – как игр, так и программ;
- наличие возможности легко реализовать логику поведения частиц (к примеру, можно сделать красивый взрыв);
- высокая производительность.

LibGDX позволяет написать код однажды и затем развертывать игру или приложение на нескольких платформах без модификации. Также в нем присутствует такая возможность, как разрабатывать приложение на основном компьютере и получать огромную выгоду быстрой разработки, вместо того чтобы ждать, когда последние изменения будут внедрены и установлены на устройство и будут скомпилированы в HTML5. Можно воспользоваться всеми инструментами языка программирования Java, для увеличения роста продуктивности, насколько это возможно.

LibGDX позволяет перейти на такой низкий уровень, какой требуется в той или иной ситуации, давая прямой доступ к файловой системе, устройствам ввода, аудио устройствам и OpenGL через единый OpenGL ES 2.0 и 3.0 интерфейс.

Наверху таких низкоуровневых возможностей создан мощный набор API, который позволяет решать общие в разработке игр задачи, такие как визуализация спрайтов, теста, построение пользовательских интерфейсов, проигрывание звуковых эффектов и музыки, линейная алгебра и тригонометрические вычисления, разбор JSON и XML, и так далее.

При необходимости, libGDX может перейти от Java к нативному коду, чтобы сосредоточиться на самой лучшей и возможной производительности. Весь этот функционал скрывается за Java API, так что можно не беспокоиться о кросс-компилировании нативного кода на всех платформах. Многие части libGDX знают специфику платформы и не нужно с ними сталкиваться.

LibGDX сосредоточен на том, чтобы являться фреймворком, нежели двигателем, признавая, что нет универсального решения. LibGDX предоставляет мощные абстракции, которые позволяют выбирать, как создавать игру или приложение.

Заключение

Кроссплатформенный Фреймворк LibGDX является отличным инструментом в создании игровых приложений благодаря своей простоте и использованию таких языков, как C и C++, для обеспечения более быстрой работы.

ЛИТЕРАТУРА

1. LibGDX // Википедия : свободная энцикл. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/LibGDX>. – Дата доступа: 25.09.2017.
2. Фреймворки // Webcreator – интернет-портал [Электронный ресурс]. – Режим доступа: https://webcreator.ru/articles/about_frameworks. – Дата доступа: 26.09.2017.
3. LibGDX – фреймворк для разработки игр // Libgdx.ru – интернет-портал [Электронный ресурс]. – Режим доступа: <http://www.libgdx.ru/2013/08/introduction.html#more>. – Дата доступа: 26.09.2017.

УДК 004.928

СОЗДАНИЕ 2D-АНИМАЦИИ ОБЪЕКТОВ

Р.А. КАРЦЕВИЧ

(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Рассматривается подход к созданию 2D-анимации объектов в игровых проектах. Показан один из способов создания анимации – способ раскадровки специально подготовленного изображения. Анимация создается за счет воспроизведения на экране кадров с определенной частотой.

Анимация играет существенную роль в повседневной жизни человека. Сегодня анимация применяется в самых различных сферах деятельности, одними из них являются:

- создание спецэффектов в кинематографе;
- создание рекламы;
- разработка компьютерных игр;
- создание анимационных фильмов;
- разработка интерфейсов сайтов и web-приложений;

Наибольшую популярность анимация получила в компьютерных играх, которая «оживляет» игру, делая ее более красочной и привлекательной, что в свою очередь привлекло огромное внимание большого количества игроков.

2D-анимация – техника, используемая для создания иллюзии движения, используя статические изображения. Анимация состоит из множества кадров, которые показываются в последовательности через определенные интервалы времени.

Рассмотрим процесс создания анимации на примере горящего огня. Анимацию горящего пламени можно достичь путем использования изображений во время полыхания пламени и бесконечным проигрыванием этих изображений в обратном порядке. На рисунке 1 показан полный цикл горения.

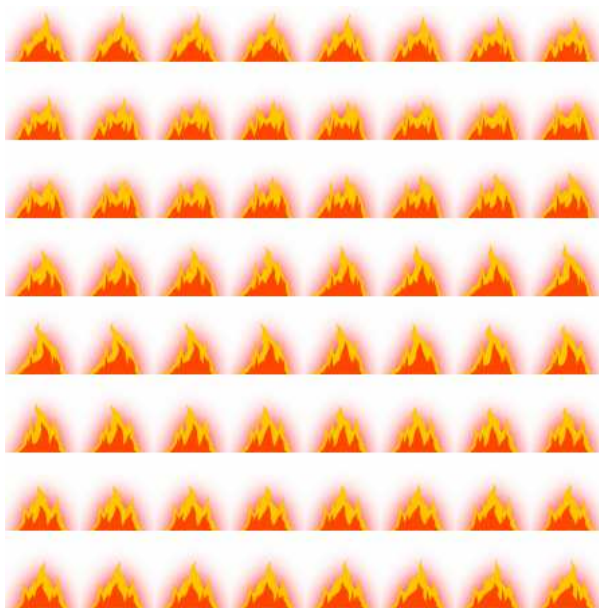


Рисунок 1. – Полный цикл горения пламени

Такое изображение называется *спрайт-листом*. Каждая область является спрайтом и называется кадром. Для создания анимации горения, спрайты должны быть нарисованы один за другим, по истечении определенного времени. В зависимости от того, как быстро должно гореть пламя, нужно определить, сколько времени кадр будет оставаться на экране. *Кадровая частота* – количество сменяемых кадров за секунду. Посмотрев на спрайт-лист, мы увидим полный цикл горения, состоящий из 64 кадров. Если полный цикл бега персонажа укладывается в одну секунду, то мы должны показывать 64 кадра в секунду. Это дает нам кадровую частоту в 64 FPS (FramesPerSecond). Двигаясь дальше, нетрудно рассчитать время состояния (время кадра), которое показывает, сколько времени кадр должен отображаться на экране перед тем, как его сменит следующий кадр: $1 \text{ секунда} / 64 = 0,064$. Другими словами, чтобы

анимация была в 64 FPS, кадр должен сменяться каждые 0,064 секунды. Анимация является простым конечным автоматом. Так, горящее пламя имеет 64 состояния, согласно спрайт-листу, представленному на рисунке 2. Пронумерованные кадры горения представляют состояние, через которое оно проходит.

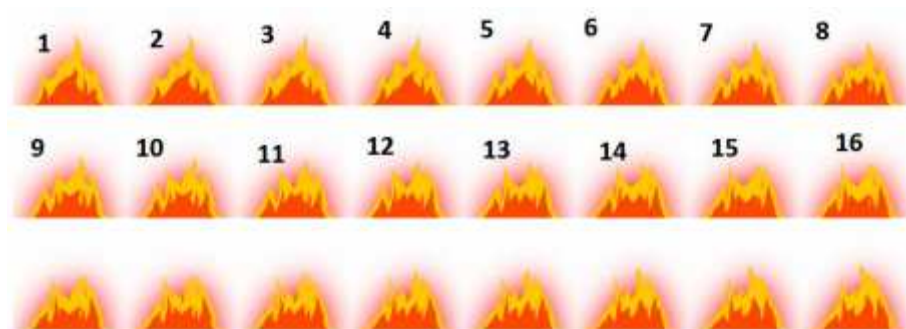


Рисунок 2. – Последовательность формирования кадров анимации

В любой момент времени конечный автомат может находиться в двух или более состояниях. Когда пламя находится в состоянии 1, то рисуется спрайт, связанный с этим состоянием. Пламя находится в этом состоянии 0,064 секунды и как только это время проходит, горение переходит в следующее состояние, которое равно 2. Это продолжается до тех пор пока не будет достигнуто последнее состояние или кадр. Циклическая анимация означает переход на начало, когда анимация достигает последнего кадра. Использовать анимацию в libGDX чрезвычайно просто. Есть одно ограничение, касающееся размера спрайт-листа, которое нужно запомнить: когда используется OpenGL 1.x, размер спрайта должен быть степенью двойки.

Заключение

В результате проведенного исследования выяснили, что одним из способов создания анимации является способ раскадровки специально подготовленного изображения. После чего с определенной частотой кадры воспроизводятся на экране, за счет этого и создается анимация.

ЛИТЕРАТУРА

1. Компьютерная анимация // Википедия – свободная энцикл. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Компьютерная_анимация. – Дата доступа: 25.09.2017.
2. Создание эффекта движения с помощью спрайтов // VirtualDub – Интернет-портал [Электронный курс]. – Режим доступа: <http://virtualdub.epublish.ru/p33aa1.html>. – Дата доступа: 26.09.2017.

УДК 004.457

**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОНТРОЛЯ ПОСЕЩАЕМОСТИ СТУДЕНТОВ
НА ОСНОВЕ АНАЛИЗА ОТПЕЧАТКОВ ПАЛЬЦЕВ****К.А. АПАНЧЁНОК****(Представлено: М.В. МАТЮШ)**

Проанализирована работа приложения для компьютера системы контроля посещаемости студентов с использованием отпечатков пальцев. Программная часть разработана на языке программирования Java для ОС Windows.

Разработанное приложение в рамках данной работы осуществляет функции, позволяющие работать с устройством и удаленной базой данных. Приложение связывает собой два этих компонента при работе с пользователем посредством отправки SQL команд на сервер БД [1] и передачи команд на устройство через COM-порт [2]. К основным функциям приложения можно приписать функции, такие как чтение/запись данных в удаленную базу данных, отправка и прием команд с устройства и вывод информации для пользователя. Все эти функции взаимодействуют между собой при работе с программой. Для более подробного анализа данных функций рассмотрим работу программы в целом.

Программа при запуске представляет собой окно с возможностью выбора основных параметров, просмотром состояния соединения сканера и базы данных. В нем по умолчанию отображена пустая таблица посещаемости, неактивные вкладки кнопок управления таблицей и меню. Меню состоит из двух пунктов: «Файл» (для авторизации и открытия базы данных) и «Параметры» (для редактирования БД администратором).

Пользователем программы является преподаватель. Для начала работы он должен открыть базу данных определенного предмета и группы. За это отвечает окно выбора основных параметров (факультет, кафедра, преподаватель, предмет, группа). При выборе параметров следующие значения сразу подгружаются в выпадающий список, исключая возможность неверного выбора с ошибкой запроса к базе данных (рис. 1).

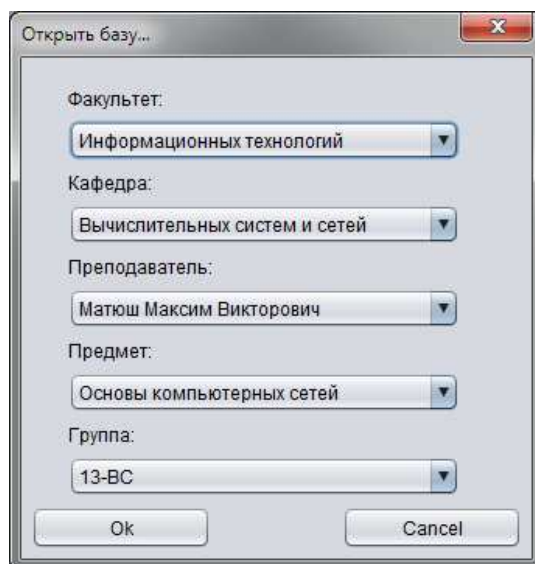


**Рисунок 1. – Окно выбора основных параметров
для начала работы**

После выбора преподавателя система проверяет на наличие у него прав администратора. Данная информация содержится в БД для каждого преподавателя (если администратор, то значение поля равно «1», если нет – «0»). Если таковы имеются, то пользователь получает возможность прекратить выбор предмета и группы для ведения журнала, а сразу же продолжить работу в режиме администратора. Иначе, после выбора всех параметров и подтверждения появляется диалоговое окно с двумя кнопками, которые уточняют тип работы с программой: режим редактирования базы данных или же режим ведения журнала. После выбора режима редактирования базы пользователю открывается доступ к пункту меню «Параметры»,

в котором собраны функции, необходимые для добавления, изменения или удаления записей в базе данных. При выборе режима ведения журнала в базу данных делается SQL запрос на получение списка студентов, списка отпечатков пальцев и списка посещения на выбранный день (если не выбран, берется текущий). Список отпечатков отправляется на устройство с порядковыми номерами студентов. В таблицу главного окна загружается список студентов выбранной группы, активируется вкладка управления посещаемостью с набором кнопок для сканирования, отметки и отправки данных на сервер (рис. 2).

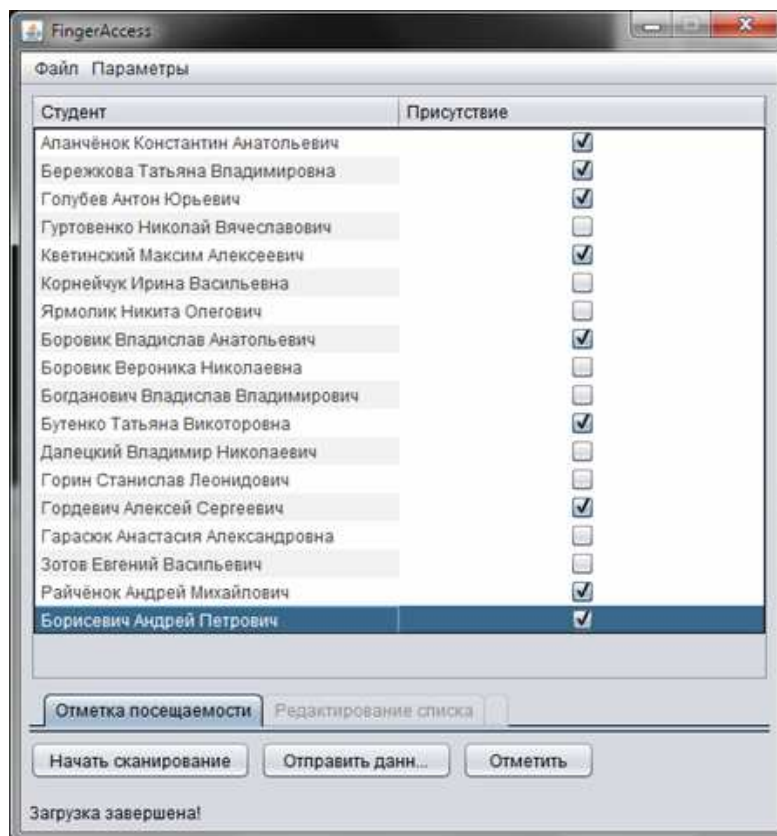


Рисунок 2. – Окно программы в режиме ведения журнала

Далее пользователь может отметить присутствующих студентов путем нажатия на кнопку «Начать сканирование» и/или «Снять/поставить присутствие». После нажатия на кнопку сканирования на устройство отправляется команда на скан и поиск отпечатка в БД сканера; кнопка меняет название на «Завершить сканирование». После этого устройство отправляет данные о нахождении совпадения в БД или нет. Если совпадение было найдено, то в таблице напротив совпадающего студента появится галочка, если нет – устройство продолжит работу. Сканирование завершается тогда, когда все студенты были отмечены, или же по нажатию пользователем кнопки «Завершить сканирование». Далее преподаватель может либо отредактировать данные вручную (если это необходимо), либо отправить данные на сервер (т.е. сохранить) нажав на соответствующую кнопку.

Преподаватель также может просмотреть посещаемость по предмету для группы за указанную дату, которая имеется в журнале. Для этого ему необходимо оторвать окно данной функции в меню и ввести значения параметров (факультет, кафедра, группа, предмет, дата). Далее в таблицу загружается список студентов и их посещаемость в выбранную дату. При необходимости, посещаемость может быть отредактирована преподавателем-администратором.

Администратор имеет расширенные возможности для управления базой данных, такие как добавление/редактирование/удаление групп, предметов, редактирование списков предметов групп. В режиме редактирования группы в таблицу загружается список студентов и шаблоны отпечатков пальцев соответственно из БД отправкой SQL запроса. Также активируется вкладка с набором функций для редактирования списка студентов. Она состоит из трех кнопок: «Добавить», «Редактировать» и «Удалить».

При добавлении студента появляется окно с двумя полями, отвечающими за ввод фамилии, имени, отчества и шаблона отпечатка пальца соответственно (рис. 3).

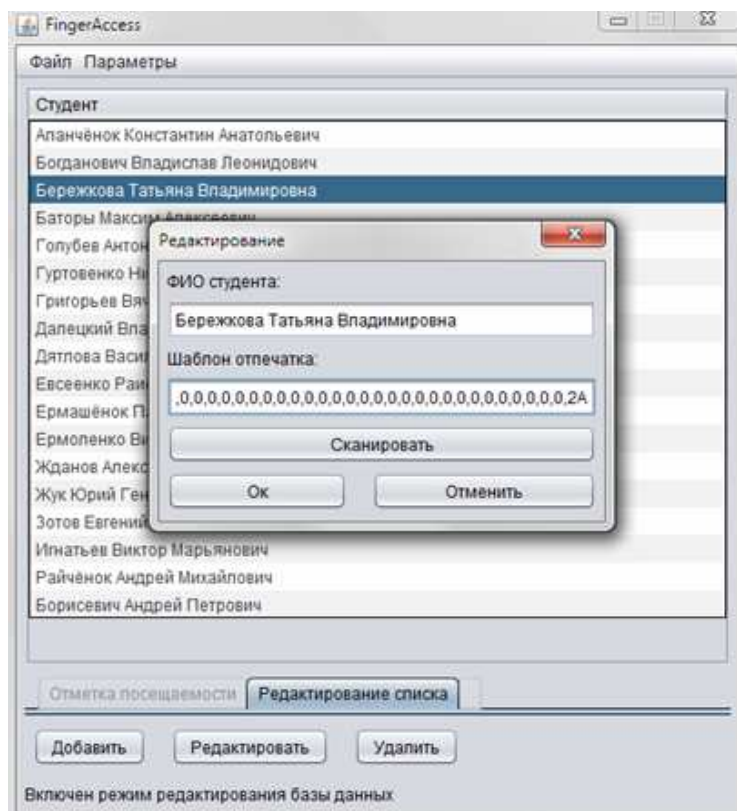


Рисунок 3. – Окно программы в режиме редактирования группы и добавления/редактирования студента

Для ввода шаблона отпечатка необходимо нажать кнопку «Сканировать». Далее программа отправляет на устройство команду сканирования пальца и отправки готового шаблона. Программа принимает данный шаблон и записывает в соответствующее поле. При сохранении данных программа формирует SQL-запрос для записи нового студента в БД с соответствующими параметрами и шаблоном отпечатка пальца. Также эти данные загружаются в таблицу приложения. При редактировании студента загружается то же диалоговое окно, только с уже загруженными данными по выбранному студенту. При удалении программа формирует SQL-запрос по удалению записи студента по заданным параметрам, а также выбранный студент удаляется из таблицы приложения.

Пользователь также может добавлять и удалять предметы из базы данных. При добавлении нового предмета пользователю предоставляется выбор, какому преподавателю привязать данный предмет, а при удалении программа удаляет все записи, включающие в себя данный предмет, такие как посещаемость и учебный план.

ЛИТЕРАТУРА

1. Примеры SQL-запросов к базе данных MySQL [Электронный ресурс]. – 2017. – Режим доступа: <http://tradebenefit.ru/primery-mysql-zaprosov>. – Дата доступа: 09.05.2017.
2. Работа с COM-портом в Java [Электронный ресурс]. – 2017. – Режим доступа: <https://ru.stackoverflow.com/questions/401234/java-%D0%A7%D1%82%D0%B5%D0%BD%D0%B8%D0%B5-%D1%81-com-%D0%BF%D0%BE%D1%80%D1%82%D0%B0>. – Дата доступа: 10.05.2017.

УДК 004.457

ОСОБЕННОСТИ АППАРАТНО-ПРОГРАММНОГО ВЗАИМОДЕЙСТВИЯ СИСТЕМЫ КОНТРОЛЯ ПОСЕЩАЕМОСТИ СТУДЕНТОВ НА ОСНОВЕ АНАЛИЗА ОТПЕЧАТКОВ ПАЛЬЦЕВ

К.А. АПАНЧЁНОК
(Представлено: М.В. МАТЮШ)

Рассматривается работа устройства системы контроля посещаемости студентов с использованием отпечатков пальцев. Представлен способ взаимодействия клиентского приложения для компьютера под управлением Windows и устройства на платформе ArduinoUno. Дано подробное описание передачи данных от приложения к устройству и обратно.

В разрабатываемом приложении и проектированном устройстве требовалось осуществить передачу данных между собой. В качестве канала передачи данных использовался последовательный СОМпорт. В Arduino имеется уже встроенный преобразователь USB-COM, поэтому работа устройство принимает и отправляет данные посредством СОМпорта [1].

Устройство по умолчанию находится в режиме ожидания команды. Всего устройство может обрабатывать набор необходимых для полноценной работы команд (табл. 1). Каждая команда предполагает дальнейшее взаимодействие со сканирующим устройством, т.е. обмен данными между Arduino и сканером отпечатков. Для того чтобы сократить время передачи и обработки команды, было решено использовать номера в качестве команд, вместо их названия.

Таблица 1. – Команды и их функции

Команда	Функция
1	Получение кода отпечатка пальца
2	Передача и запись кода отпечатка пальца в сканер
3	Сканирование пальца и поиск совпадения
4	Очистка БД сканера

При выполнении *первой команды* на сканер отправляется команда считывания отпечатка пальца и записи результата в первый буфер сканера, затем снова команда считывания отпечатка пальца и запись во второй буфер. После отправляется команда на создание усредненного шаблона отпечатка (для того, чтобы ввести погрешность при последующем поиске совпадений с данным отпечатком) и в конце – команда передачи шаблона со сканера на Arduino и команда передачи шаблона с Arduino в приложение.

При выполнении *второй команды* Arduino ожидает приема кода отпечатка пальца определенной длины и номера для записи шаблона в память сканера. Затем Arduino отправляет команду передачи шаблона на сканер и номера для записи в необходимую ячейку памяти.

При выполнении *третьей команды* на сканер отправляется команда для сканирования отпечатка, записи его в буфер и поиск совпадения в базе сканера. Сканер, в свою очередь, отправляет обратно результат, который потом дублируется на отправку в приложение.

При выполнении *четвертой команды* на сканер отправляется команда, которой устройство очищает энергонезависимую память сканера, а также временные буферы.

Таким образом, пользовательское приложение отправляет команду и ожидает приема метасимволов, означающих выполнение команды, а затем продолжает работу с пользователем (псевдокод 1).

Псевдокод 1 – функция отправки команды поиска совпадения

```
String findMatch() {
    String ret = null;
    try {
        data2 = "";
        serialPort.writeString("3\n");    //отправкакоманды 3
        while (true) {sleep(1);
            data = serialPort.readString();    //чтениеответныхданных
            if (scan == 0) { break;}
            if (data != null) {data2 += data;}
```

```

    if (data2.contains("$#")) {          //проверка на наличие мета-символов
    int a = data2.indexOf("ID=") + 3;
    int b = data2.indexOf("$");
    ret = data2.substring(a, b); //получениеномерасовпадения
    break; }
    if (data2.contains("Did not find a match")) { break;} //проверкананесовпадение
}
} catch (SerialPortException ex) { Logger.getLogger(Window.class.getName()).log(Level.SEVERE,
null, ex);
} catch (InterruptedException ex) {Logger.getLogger(Window.class.getName()).log(Level.SEVERE,
null, ex);}
data2 = ""; //очистка буфера приема данных
returnret;} //вывод результата функции

```

На устройстве Arduino обрабатываемая команда выполняется и выдает результат как при нахождении совпадения (рис. 1), так и при его отсутствии (рис. 2).

```

FingerprintTest...#
Success!#
SWITCH MODE#
SWITCH = 3
MatchMode#
Scan finger#
Taked#
Converted#
Found a print match!#
Found ID=1$#
SWITCH MODE#

```

Рисунок 1. – Результат выполнения команды поиска совпадений при нахождении отпечатка в БД

```

SWITCH MODE#
SWITCH = 3
MatchMode#
Scan finger#
Taked#
Converted#
Did not find a match#
SWITCH MODE#

```

Рисунок 2. – Результат выполнения команды поиска отпечатка при отсутствии совпадения

Разработанное устройство выполнено на базе аппаратной платформы ArduinoUno и сканера отпечатков пальцев ZFM-20 [2]. Данный сканер позволяет получать шаблоны отпечатков за время < 1 секунды, хранить ввстроенной памяти до 1600 значений. Аппаратная часть представляет собой комплекс из двух этих устройств, которые взаимодействуют между собой и связываются с программой на ПК. Связь между ПК и устройством осуществлена посредством COM-порта [3]. Сканер по умолчанию настроен на прием и передачу информации в шестнадцатиричном коде путем формирования пакетов данных (табл. 2).

Таблица 2. – Формат построения пакета данных

Заголовок (2 байта) Обычно значение EF01H	Адрес (4 байта)	Идентификатор пакета данных (1 байт) 01H – командный; 02H – пакет данных; 07H – ответный пакет; 08H – завершающий пакет данных	Размер пакета (2 байта)	Содержание пакета (размер задается в настройках сканера. Максимум 256 байт)	Контрольная сумма (2 байта)

Для корректной и более удобной работы со сканером использована библиотека AdufruitFingerPrintLib [4], которая позволяет использовать большинство функций данного сканера без формирования пакетов данных вручную. На Arduino реализованы функции, необходимые для обмена данными с ПК и сканером.

При включении Arduino происходит авторизация сканера (инициализация порта обмена данными со сканером и проверка его на готовность работы) и подключения с ПК (открытие порта для обмена данными с ПК), затем идет ожидание получения одной из команд от ПК, перечисленных выше. Рассмотрим каждую функцию подробнее в рамках устройства.

Функция получения шаблона отпечатка пальца при выполнении отправляет на сканер команду начала сканирования и ожидает сканирования отпечатка. После того как палец был просканирован, его скан загружается в один из двух буферов сканера для дальнейшей обработки. Затем сканер снова переходит в режим сканирования и записывает второй скан в другой буфер. Это необходимо для построения усредненного значения шаблона отпечатка пальца. Такой шаблон будет распознаваться сканером даже при неполном соответствии (70–80%), не допуская при этом ошибок при поиске отпечатка в БД. После составления шаблона на сканер подается команда для выгрузки шаблона со сканера на Arduino, которая, в свою очередь, отправляет его через COM-порт на ПК (псевдокод 2).

Псевдокод 2 – функция получения шаблона отпечатка пальца со сканера

```
int cas1(){
  Serial.println("GetCharFromIMG mode");
  Serial.println("Scan finger#");
  int p = -1;
  p = GetImage(1); //отправка команды получения скана пальца
  if(p != -1){
    delay(2);
    p = ImageToTz(1); //конвертирование изображения в массив символов и запись в буфер №1
    delay(1000);
    while (p != FINGERPRINT_NOFINGER) {
      p = finger.getImage(); //отправка команды получения скана пальца для проверки на наличие
//пальца на сканере (проверка ввода)
    }
    p = GetImage(1); //отправка команды получения скана пальца
    if(p != -1){
      delay(2);
      p = ImageToTz(2); //конвертирование изображения в массив символов и запись в буфер № 2
      p = createModel(); //создание усредненного шаблона отпечатка пальцев при совпадении
//более 90% (то есть при скане одного и того же пальца) и запись результата в буфер № 1
    }
    if(p == 0)
    {
      downloadModel(1); //если создание успешно, то отправляется команда получения значения
//из буфера № 1
    }
    else{
      Serial.println("NOTMATCH"); } } //иначе, вывод сообщения «НЕТ СОВПАДЕНИЯ»
```

Функция добавления шаблона отпечатка во внутреннюю память устройства при вызове запускает прием данных шаблона отпечатка пальца с ПК, затем принимает уникальный идентификатор данного отпечатка. Затем запускается функция отправки шаблона отпечатка в буфер сканера. После выполняется функция записи данного отпечатка в память по указанному ID (псевдокод 3).

Псевдокод 3 – функция добавления шаблона отпечатка в память

```
int cas2(void){
  Serial.println("DownloadCharToCharbuf mode");
  uint8_t packet[688];
  Serial.println("Start recive char...#");
  int p = 0;
  p = reciveChar(packet); //получение данных с COM-порта от ПК
  Serial.println("End recive#");
```

```

Serial.println("Enter ID#");
int id = readnumber();    //получениеидентификатора
Serial.print("ID=");
Serial.println(id);
Serial.println("Upload#");
uploadModel(1, packet);   //передачашаблонаотпечаткапальцавбуфер №1
Serial.println("Storing#");
storeModel(id);           //записьзначениябуфера №1 впамятьподномеромid
}

```

Функция поиска совпадений по скану при выполнении отправляет команду захвата отпечатка пальца со сканера. Затем при наличии значения пальца конвертируется в буфер № 1, и отправляется команда получения номера совпадения (псевдокод 4).

Псевдокод 4 – поиск совпадения в памяти устройства

```

int cas3(void){
Serial.println("MatchMode#");
Serial.println("Scan finger#");
int p = -1;
p = GetImage2(); //получение скана пальца
if(p != -1){
delay(2);
p = ImageToTz(1);    //конвертирование изображения в символьное представление
getMatch();           //поиск совпадения о сходстве отпечатка с уже имеющимися в памяти
}
}

```

Функция очистки памяти сканера предназначена для очистки шаблонов перед загрузкой новых шаблонов какой-либо группы (псевдокод 5).

Псевдокод 5 – функция очистки памяти сканера

```

intcas4(void){
Serial.println("EmptyDataBase");
emptyBase();
}

```

ЛИТЕРАТУРА

1. Работа с COM-портом Arduino [Электронный ресурс]. – 2017. – Режим доступа: <http://arduino.ru/forum/programirovanie/rabota-s-com-portom>. – Дата доступа: 10.05.2017.
2. ZFM-20dingerprintmoduledatasheet [Электронный ресурс]. – 2017. – Режим доступа: https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library/blob/master/documentation/ZFM-20_Fingerprint_Module.pdf. – Дата доступа: 10.05.2017.
3. Работа с COM-портом в Java [Электронный ресурс]. – 2017. – Режим доступа: <https://ru.stackoverflow.com/questions/401234/java-%D0%A7%D1%82%D0%B5%D0%BD%D0%B8%D0%B5-%D1%81-com-%D0%BF%D0%BE%D1%80%D1%82%D0%B0>. – Дата доступа: 10.05.2017.
4. Arduino library for interfacing to the fingerprint sensor [Электронный ресурс]. – 2017. – Режим доступа: <https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library/>. – Дата доступа: 10.05.2017.

УДК 681.5:316.7(476)

**ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ
СИСТЕМЫ УПРАВЛЕНИЯ АФИШЕЙ КУЛЬТУРНЫХ СОБЫТИЙ РЕСПУБЛИКИ БЕЛАРУСЬ****В.Ю. ДЕМЕНТЬЕВ****(Представлено: Ю.Н. КРАВЧЕНКО)**

Демонстрируется разработка системы управления афишей культурных событий Республики Беларусь. Показано создание системы для удобного управления афишей культурных событий Республики Беларусь. Рассматриваются вопросы, связанные с процессом проектирования базы данных системы управления.

Любое приложение, будь то web-приложение, desktop или мобильное, которое нацелено на предоставление актуальной информации, нуждается в администрировании, т.е. управлении контентом в удобной для администратора форме. Обычно для таких потребностей используются административные панели, которые позволяют обращаться к базе данных посредством web-форм, изменять содержимое приложений в визуальных редакторах и т.д. Поэтому основное преимущество разрабатываемой системы управления в том, что администратору не нужно редактировать базу данных напрямую, а из-за необходимости хранения множества различных сущностей и большого объема данных проектируемая база данных должна быть наиболее простой, нормализованной и эффективной.

Для определения сущностей, хранимых в базе данных, необходимо выделить функции, которые будут доступны в системе управления. *Во-первых*, стоит отметить, что система будет разделена на две части: 1) *глобальная система управления* (для управления и моделирования мест проведения событий (театры и кинотеатры, дома культуры, открытые концертные площадки и т.д.), для управления администраторами и т.п.); 2) *система управления афишей места проведения культурного мероприятия* (для непосредственного управления афишей). *Во-вторых*, из-за разделения системы будет выделено две роли – глобальный администратор и администратор проведения мероприятия.

Таким образом, *администратору по месту проведения* будет доступен небольшой ряд функций по работе с афишей событий в непосредственно выделенном для них месте и редактированию информации о самом месте:

- добавление/редактирование/удаление мероприятий (событий);
- добавление/редактирование/клонирование/удаление цен;
- добавление/удаление сеансов проведения мероприятий;
- редактирование информации о месте проведения событий.

Глобальный администратор будет иметь намного больше прав:

- добавление/редактирование/удаление мероприятий для любого места проведения события;
- добавление/редактирование/клонирование/удаление цен для любого места проведения события;
- добавление/удаление сеансов для любого места проведения события;
- редактирование информации о любом месте проведения мероприятия;
- редактирование информации любого администратора;
- добавление новых мест для проведения событий в систему;
- регистрация новых администраторов для мест проведения мероприятий;
- просмотр лога изменений любого администратора;
- черный список администраторов;
- просмотр различной статистики по местам проведения событий.

С учетом выдвинутых к системе управления функций, можно выделить следующие *сущности проектируемой базы данных*:

- администратор (user) – пользователь системы управления, обладающий определенными правами редактирования данных места проведения мероприятий;
- место проведения события (place) – заведение, в котором проводятся сеансы мероприятий;
- город (city) – город, в котором происходит событие;
- мероприятие (action) – событие, происходящее в назначенное время и в назначенном месте;
- тип события (type) – тип мероприятия (кинофильм, концерт и любые другие массовые события);
- сеанс (session) – единица проведения события. Информация о сеансе включает в себя стоимость сеанса, дату и время действия мероприятия;
- цена (cost) – стоимость одного сеанса;
- архив (archive) – список событий, которые завершились;
- лог (log) – запись об изменении в системе, которые внесены администратором места проведения событий.

Чтобы построить схему реляционной базы данных, необходимо определить совокупность отношений, которые составляют базу данных. Необходимо, чтобы выполнялось условие целостности отношений: в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа, т.е. любая запись любого отношения должна быть отличной от любой другой записи этого отношения.

В проектируемой базе данных все таблицы будут иметь первичные ключи и связаны между собой связью один ко многим.

Схема спроектированной базы данных, нормализованной до 3-ей нормальной формы, представлена на рисунке 1.

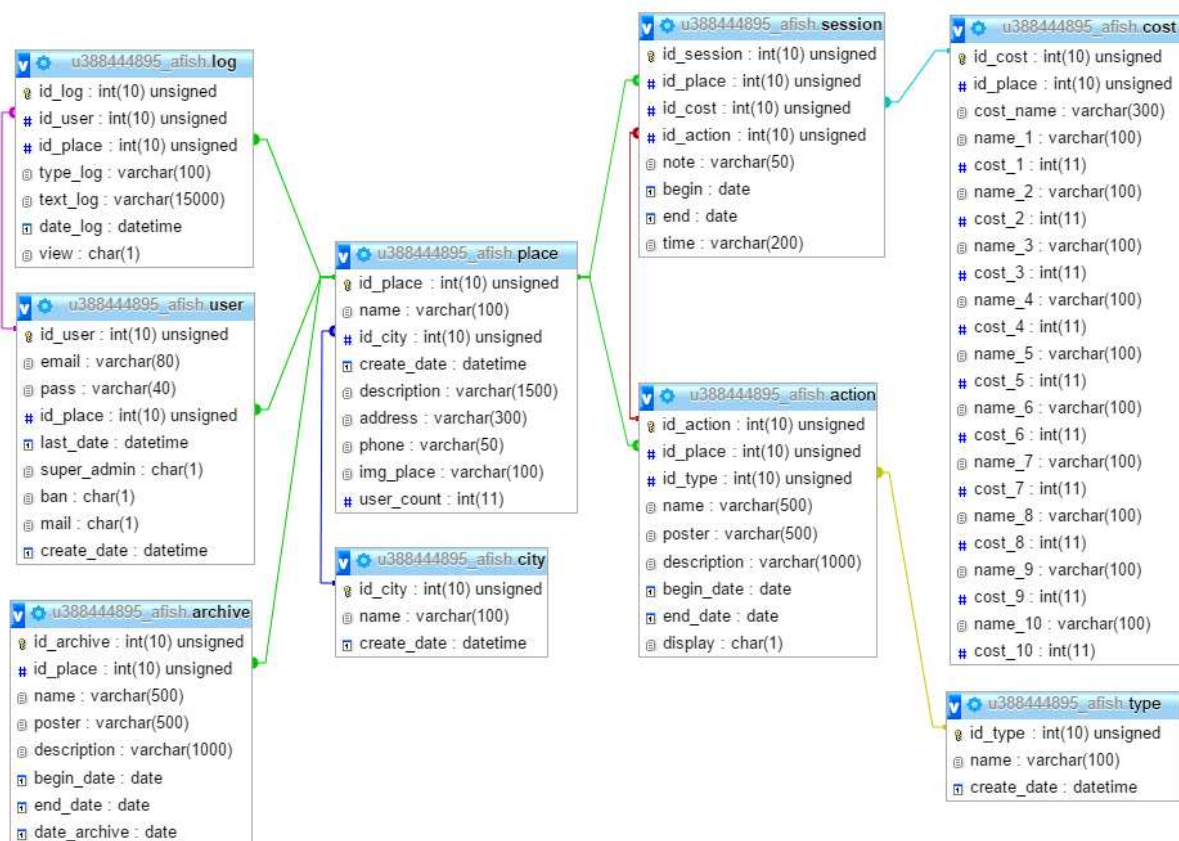


Рисунок 1. – Схема спроектированной базы данных

В качестве СУБД была выбрана MySQL. СУБД MySQL обладает высокой скоростью работы по сравнению с другими системами управления. Также преимуществом MySQL является то, что это многопользовательская система, она не налагает ограничений на количество пользователей, одновременно работающих с базой данных.

MySQL может хранить огромное количество записей в своих таблицах. Этим обеспечивается возможность создания крупных проектов. Немаловажным показателем является также простая и эффективная система безопасности MySQL.

Для удобного формирования архива событий необходимо создать триггер (after delete) на удаление мероприятия.

Заключение. В результате проделанной работы спроектирована база данных, при помощи которой можно создать систему управления афишей культурных событий для удобной работы администратора. Также использование базы данных делает удобным хранение и управление данными афиши.

ЛИТЕРАТУРА

1. Кузнецов, М.В. MySQL 5 / М.В. Кузнецов, И.В. Симдянов. – СПб. : БХВ-Петербург, 2010. – 1024 с.
2. Малыгина, М. Базы данных: основы, проектирование, использование / М. Малыгина. – М. : ВНУ, 2004. – 512 с.
3. Фестиваль педагогических идей «Открытый урок» [Электронный ресурс] / ИД «Первое сентября». – М., 2014. – Режим доступа: <http://festival.1september.ru>. – Дата доступа: 29.03.2014.

УДК 316.7:316.7(476)

**ПРОЕКТИРОВАНИЕ КЛИЕНТСКОЙ ЧАСТИ СИСТЕМЫ УПРАВЛЕНИЯ
АФИШЕЙ КУЛЬТУРНЫХ СОБЫТИЙ РЕСПУБЛИКИ БЕЛАРУСЬ****В.Ю. ДЕМЕНТЬЕВ,****(Представлено: Ю.Н. КРАВЧЕНКО)**

Рассматриваются вопросы, связанные с процессом проектирования клиентской части системы управления афишей культурных событий Республики Беларусь. Цель – создание системы для удобного управления афишей культурных событий. Использование спроектированной системы позволит облегчить процесс управления афишей любых культурных событий и мероприятий.

Цель разработки клиентской части системы управления, в первую очередь, состоит в удобстве работы администратора с записями из базы данных и исключение возможных ошибок при работе с базой данных. Таким образом, особое внимание необходимо уделить разработке графического пользовательского интерфейса.

Если рассматривать системы управления в целом, то можно выделить несколько минусов. Как правило, все CMS разрабатываются под какие-то общие потребности (добавление статей, модерирование пользователей и др.) и возникают определенные трудности при необходимости добавления каких-либо специфических и уникальных для таких CMS модулей. Из-за этого вся система становится более громоздкой. Некоторые возможности системы управления не используются, так как они не нужны для текущих потребностей пользователя, но на них тратятся определенные ресурсы. Кроме этого, стиль таких CMS недостаточно современен. На данный момент Google задает тенденцию в так называемом Material Design, схожим со стилем минимализм. В проектируемой клиентской части системы управления будет использован данный стиль, так как он все больше набирает популярность и выглядит достаточно строго и аккуратно. Нет необходимости в красочных изображениях, из-за которых пользователь может теряться в системе и не воспринимать или упускать важную информацию. Вся панель администратора должна быть выполнена в едином приятном для глаза стиле. Также стоит отметить, что многие системы управления не всегда удобны. Для выполнения основных задач бизнес-процесса необходимо выполнить неочевидные шаги для пользователя. Разрабатываемая система должна быть совершенно понятна и прозрачна, чтобы любой администратор места проведения культурного мероприятия мог интуитивно выполнить любое действие путем нажатия не более двух кнопок. Отдельное внимание будет уделено эргономике. Удобное размещение элементов также позволит облегчить работу администратора и снизить когнитивную нагрузку.

В качестве среды разработки был выбран JetBrais PhpStorm 8, так как данная среда разработки имеет отличный интерфейс и поддержку самых последних версий PHP. Кроме этого, PhpStorm предоставляет отличные возможности для удобной разработки на языках JavaScript, HTML и CSS.

За основу разработки дизайна была взята стилистика CSS библиотеки Skeleton. Данная библиотека предоставляет минимальные возможности для начала разработки собственного проекта, поэтому все основные стили будут написаны собственноручно.

Для отображения иконок в приложении будет использоваться библиотека Font Awesome. Это векторные иконки, которые интегрированы в шрифт. Их преимущество в том, что они способны изменять размер без потери качества, а также к ним можно применять различные CSS стили, например, изменение цвета.

Для удобной работы с JavaScript будет использоваться библиотека JQuery. Она предоставляет простой доступ к DOM-элементам страницы. Большинство POST/GET запросов при работе с данными в панели управления будут обрабатываться посредством технологии AJAX. Таким образом, это уменьшит время ожидания получения ответов от сервера, так как подгружаться на странице будут только необходимые данные, остальные данные затрагиваться не будут.

Для построения графической составляющей необходимо определить, какие функции должна выполнять система управления. Прежде всего, отметим, что система будет разделена на две части – *глобальную систему управления* (для управления и моделирования мест проведения событий (театры и кинотеатры, дома культуры, открытые концертные площадки и т.д.) и *систему управления афишей места проведения культурного мероприятия* (для непосредственного управления афишей).

Из-за разделения системы выделено две роли – *глобальный администратор* и *администратор проведения мероприятия*.

Администратору по месту проведения будет доступен небольшой ряд функций по работе с афишей событий в непосредственно выделенном для них месте и редактированию информации о самом месте:

- добавление/редактирование/удаление мероприятий (событий);

- добавление/редактирование/клонирование/удаление цен;
- добавление/удаление сеансов проведения мероприятий;
- редактирование информации о месте проведения событий;

Глобальный администратор будет иметь намного больше прав:

- добавление/редактирование/удаление мероприятий для любого места проведения события;
- добавление/редактирование/клонирование/удаление цен для любого места проведения события;
- добавление/удаление сеансов для любого места проведения события;
- редактирование информации о любом месте проведения мероприятия;
- редактирование информации любого администратора;
- добавление новых мест для проведения событий в систему;
- регистрация новых администраторов для мест проведения мероприятий;
- просмотр лога изменений любого администратора;
- черный список администраторов;
- просмотр различной статистики по местам проведения событий.

Все страницы будут сверстаны по единому шаблону: шапка – основной контент – подвал.

Прежде чем начать работу с системой управления, необходимо пройти процедуру авторизации. Это выполняется посредством формы входа, которая состоит из двух полей для ввода e-mail адреса и пароля, а также кнопки «Войти».

Далее будут описаны страницы, доступные глобальному администратору.

В шапке сайта располагается меню, в котором находятся следующие пункты меню:

- статистика;
- управление;
- лог изменений;
- выход.

В свою очередь, пункт меню «Управление» может быть раскрыт в виде выпадающего списка, который включает в себя следующие пункты:

- управление местами проведения событий;
- управление пользователями;
- управление форматами мероприятий;
- управление списком городов.

На странице статистики будет расположена различная информация по количеству сущностей в системе, статистика по местам проведения мероприятий и т.п.

На странице «Лог изменений» будет расположен список изменений, которые внесли в систему администраторы мест проведения событий. Глобальный администратор сможет следить за всеми изменениями и принимать меры в случае нарушения каких-либо правил.

Данные страницы состоят из списка сущностей, некоторой информации о них и действий, которые для них возможны. Используя данную страницу, администратор может добавить новое место проведения мероприятия в систему, отредактировать или удалить существующее, либо войти в любое из мест событий от лица другого администратора и выполнять любые действия, доступные обычному администратору проведения событий. На странице управления пользователями администратор также может добавлять новых администраторов, удалять или редактировать существующих, а также добавлять их в черный список.

Страницы управления форматами фильмов и списком городов также похожи. На них отображается список сущностей с возможностью редактирования и удаления, а также дополнительное поле быстрого добавления новой сущности. Все действия выполняются посредством технологии AJAX.

Так как у администратора места проведения события другие функции, в его панели управления располагаются иные пункты меню:

- афиша;
- добавить;
- архив;
- в панель управления;
- настройки;
- выход.

При этом пункт меню «Добавить» включает в себя следующие подпункты, которые раскрываются в виде выпадающего списка:

- добавить событие;
- добавить сеансы;
- добавить цены.

На странице «Афиша» располагается список текущих событий, которые проводятся в текущем месте, а также его сеансы и возможные действия. На данной странице администратор может сделать ме-

роприятие невидимым для конечных пользователей, перейти к редактированию события или удалить его. После удаления событие помещается в архив и весь список удаленных мероприятий доступен на соответствующей странице. В архиве администратор может удалить событие полностью из системы.

Пункт меню «В панель управления» отображается только, если в панель управления вошел глобальный администратор через страницу «Управление местами проведения событий». Она возвращает администратора обратно к списку мест проведения событий, зарегистрированных в системе. Для обычного администратора данный пункт недоступен.

На странице настроек администратор может изменять данные о месте проведения событий – телефон, адрес или описание. Также там можно сменить пароль.

Страницы добавления и редактирования событий похожи и состоят из одинаковых полей. На них есть возможность загрузки рекламного изображения для планируемого события и выбор формата мероприятия. Кроме этого на данных страницах указывается подробное описание проводимого события.

На странице добавления цен располагается список уже добавленных цен в систему и форма для добавления новой цены. При этом цены могут быть составные, т.е. одна запись может содержать до 10 цен. Таким образом, если на месте проведения мероприятия существует разделение зала на сектора, с разной стоимостью, администратор в пределах одной записи может указать стоимость каждого сектора. На той же странице администратор может удалить, отредактировать или создать клон записи.

На странице добавления сеансов располагаются выпадающие списки с доступными событиями и ценами, которые необходимо выбрать для добавления сеанса. Там же необходимо указать период и время сеанса.

Кроме описанных пунктов меню, в системе в подвале каждой страницы доступны следующие пункты меню:

- помощь;
- правила системы;
- написать администратору.

На странице помощи располагается список возможных проблем и методы их решения в виде видеозаписи, а также примеры использования всех доступных администратору функций. На странице правил системы располагается список правил, из-за невыполнения которых администратор может попасть в черный список системы. А при нажатии ссылки «Написать администратору» пользователю отобразится всплывающее окно с формой для отправки сообщения администратору.

Заключение. Интерфейс является понятным пользователю без обучения работы с системой управления афишей. Присутствует очень подробная страница помощи в случае, если у пользователя возникли трудности. В случае выполнения неверных действий пользователю будет показано сообщение об ошибке, а в случае корректного выполнения – сообщение об успешном завершении операции.

Использование спроектированной системы позволит облегчить процесс управления афишей любых культурных событий и мероприятий.

ЛИТЕРАТУРА

1. Спецификация Google – Material design [Электронный ресурс] / Google. Material design. – Режим доступа: <http://www.google.com/design/spec/>. – Дата доступа: 29.09.2015.
2. Макфарланд, Д. Большая книга CSS3 / Д. Макфарланд. – 3-е изд. – СПб. : Питер, 2014. – 608 с.

УДК 004.624

ИСПОЛЬЗОВАНИЕ LIQUIBASE
ДЛЯ УПРАВЛЕНИЯ ИЗМЕНЕНИЯМИ СХЕМЫ БАЗЫ ДАННЫХ

Д.Ю. ДЁКИН

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматривается платформно-независимая библиотека с открытым исходным Liquibase, при помощи которой можно осуществлять контроль за изменениями схемы базы данных. Проанализированы некоторые возможности библиотеки Liquibase. Представлены базовые примеры, демонстрирующие простоту применения данной технологии в рамках управления изменениями структуры базы данных.

При разработке большинства приложений используются различные системы контроля версий (Git, Subversion и т.п.). Возникает вопрос, как с течением времени меняется схема БД, как это отслеживается, чем управляется? Случается, что изменения базы данных на тестовых и «боевых» серверах делаются вручную. Данный подход может не вызывать сложностей, если над проектом работает один или два разработчика, но, если работает целая команда, становится трудно синхронизировать все изменения между разработчиками.

Liquibase – платформно-независимая библиотека с открытым исходным кодом (реализована на Java), которая предназначена для управления изменениями схемы БД, а также является хорошим инструментом в миграции с одной базы данных на другую.

Чтобы начать работать с *Liquibase*, требуется указать несколько настроек для библиотеки:

1. Драйвер базы данных.
2. Путь к драйверу.
3. Url к базе данных.
4. Логин и пароль к базе данных.
5. Путь к чейнджлогу.

Основными понятиями, которыми оперирует *Liquibase*, являются:

1. Чейнджсет (от англ. changeset – набор изменений).
2. Чейнджлог (от англ. changelog – файл, состоящий из набора чейнджсетов).

Таким образом, отдельный чейнджлог представляет собой совокупность чейнджсетов в рамках какой-то отдельно взятой таблицы базы данных.

Код чейнджсетов может быть описан при помощи XML, JSON, YAML, SQL или даже на groovy. Например, при помощи следующего кода (листинг 1) можно создать таблицу **person** с двумя полями: *id*, *name*.

Листинг 1 – Создание таблицы «person»

```
<databasechangelog
  xmlns=http://www.liquibase.org/xml/ns/dbchangelog xmlns:xsi=http://www.w3.org/2001/XMLSchema-
instance xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-2.0.xsd">
  <changeset id="2017-09-09-00" author="dzmitry@mail.com">
    <createtable tablename="person">
      <column name="id" type="BIGINT" autoincrement="true">
        <constraints nullable="false" primarykey="true"/>
      </column>
      <column name="name" type="VARCHAR(255)">
        <constraints nullable="false"/>
      </column>
    </createtable>
  </changeset>
</databasechangelog>
```

Liquibase имеет встроенную поддержку rollback (отмены) для некоторых типов чейнджсетов, как, например, для чейнджсета выше (*createTable*). Другие чейнджсеты автоматически быть откатенными не могут. В таких случаях предусмотрена секция *rollback*, позволяющая описывать действия для отмены изменений чейнджсета (листинг 2).

Листинг 2 – Пример чейнджсета с секцией «rollback»

```
<databasechangelog
  xmlns=http://www.liquibase.org/xml/ns/dbchangelog xmlns:xsi=http://www.w3.org/2001/XMLSchema-
instance xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-2.0.xsd">
  <changeset id="2017-09-09-00" author="dzmitry@mail.com">
    <insert tablename="person">
      <column name="id" value="John Smith"/>
    </insert>
    <rollback>
      DELETE FROM person WHERE name LIKE 'John Smith';
    </rollback>
  </changeset>
</databasechangelog>
```

Чейнджсеты в *Liquibase* определяются уникальностью комбинации тегов `id`, `author`, а также имени файла чейнджлога. Таким образом, чтобы успешно применять чейнджсеты (в рамках одного чейнджлога), необходимо задавать уникальное поле для `id` и `author`.

На практике же (чаще всего) в качестве `id` берется значение текущей даты, и в конец добавляется порядковый номер чейнджсета. Поэтому если бы в чейнджлог из листинга 2 нужно было бы добавить еще один чейнджсет, то его `id` имело бы значение «2017-09-09-01».

Имея несколько чейнджлогов, можно создать один кумулятивный файл, который будет импортировать другие чейнджлоги (листинг 3).

Листинг 3 – Пример кумулятивного чейнджлога

```
<databasechangelog
  xmlns=http://www.liquibase.org/xml/ns/dbchangelog>
  <include file="v-1.0/person-v-0-1.xml" relativeToChangelogFile="true"/>
  <include file="v-1.0/data-v-0-1.xml" relativeToChangelogFile="true"/>
</databasechangelog>
```

Используя *Liquibase*, стоит следовать ряду правил и рекомендаций, который облегчит управление версиями схемы базы данных в дальнейшем. Итак, версионность приложения должна быть отражена в структуре папок чейнджлогов:

1. Чейнджлоги изменения базы данных должны содержаться в отдельных папках, соответствующих версиям приложения
2. Держать кумулятивный файл чейнджлогов для каждой версии.
3. Соблюдать правила именования чейнджлогов.
4. Лучше много маленьких чейнджсетов, чем один большой.
5. Чейнджлоги, изменяющие схему базы данных, должны быть атомарными.

Liquibase оперирует значениями хеша чейнджсета. Поэтому если чейнджсет уже был применен, но оказалось, что нужно добавить что-то еще, то должен создаваться отдельный чейнджсет. Иначе, если отредактировать имеющийся чейнджсет, изменится хеш данного чейнджсета. В таком случае *Liquibase* не применит данные изменения к базе данных.

Заключение

Рассмотрены некоторые возможности библиотеки *Liquibase*, а также даны советы и рекомендации по организации и хранению чейнджсетов. Предоставлены базовые примеры, демонстрирующие простоту применения данной технологии в рамках управления изменениями структуры базы данных.

ЛИТЕРАТУРА

1. Хабрахбр [Электронный ресурс] Использование Liquibase без головной боли. 10 советов из опыта реальной разработки. – Режим доступа: <https://habrahabr.ru/post/179425>. – Дата доступа: 24.09.2017.
2. Хабрахбр [Электронный ресурс] Управление миграциями БД с Liquibase. – Режим доступа: <https://habrahabr.ru/post/179425>. – Дата доступа: 24.09.2017.
3. Ezcode [Электронный ресурс] Версионирование структуры БД с помощью Liquibase. – Режим доступа: <http://easy-code.ru/lesson/database-versioning-liquibase>. – Дата доступа: 24.09.2017.

УДК 004.657

МУВАТИС КАК ИНСТРУМЕНТ ДЛЯ РАБОТЫ С БАЗАМИ ДАННЫХ
В JAVA-ПРИЛОЖЕНИЯХ

Д.Ю. ДЁКИН

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматривается MyBatis фреймворк для отображения данных из базы на доменные объекты в Java-приложениях. Представлен краткий обзор базовых возможностей MyBatis фреймворка и его базовой конфигурации. Приведены примеры маппингов и запросов к базе данных, вызова запросов из Java-кода.

При проектировании и разработке приложений практически всегда приходится иметь дело с базами данных. Поэтому встает вопрос о выборе того или иного ORM или persistence фреймворка, который будет служить для отображения данных из базы в пользовательские объекты. Существует достаточно большое количество различных ORM, таких как Hibernate, JPA, EclipseLink и др. Каждый из них имеет свои достоинства и недостатки. Большинство фреймворков поддерживают JPA спецификацию.

Рассматриваемый в данной статье фреймворк является альтернативой JPA спецификации.

MyBatis является persistence фреймворком, который отображает SQL запросы на доменные объекты. Данный фреймворк (так же, как и ORM фреймворки) ликвидирует необходимость писать шаблонный и ненужный код для уже давно решенных задач. MyBatis позволяет отображать данные из базы в примитивные Java-типы и POJO (Plain Old Java Object, простой Java-объект). MyBatis может быть сконфигурирован как при помощи XML, так и при помощи аннотаций. Для удобства и простоты последующие примеры будут приводиться на XML.

MyBatis хорошо подходит для тех случаев, когда в приложении имеется достаточно много различных сложных выборок из нескольких таблиц (много JOIN-ов). Все дело в том, что MyBatis позволяет отображать SQL-запросы на сущности. Поэтому производительность запроса полностью определяется разработчиком. Это позволяет полностью контролировать, какие запросы и как именно будут отправляться в БД.

Для базовой конфигурации необходимо указать источник данных (от англ. data source) с именем драйвера и url к БД, логином и паролем. Также в конфигурационном файле указываются type alias (доменные объекты), mapper (xml-файлы, которые хранят SQL-запросы). Пример приведен в листинге 1.

Листинг 1 – Пример конфигурационного файла

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <properties resource="config.properties">
    <property name="schema" value="db_schema"/>
  </properties>

  <dataSource type="POOLED">
    <property name="driver" value="${driver}"/>
    <property name="url" value="${url}"/>
    <property name="username" value="${username}"/>
    <property name="password" value="${password}"/>
  </dataSource>

  <typeAliases>
    <typeAlias type="by.psu.domain.User"/>
    <typeAlias type="by.psu.domain.Role"/>
  </typeAliases>

  <mappers>
    <mapper resource="by/psu/repository/IUserMapper.xml"/>
    <mapper resource="by/psu/repository/IRoleMapper.xml"/>
  </mappers>
</configuration>
```

Для того чтобы показать, как происходит отображение результатов выборки на доменные объекты, введем следующие простые типы: User, Role (методы доступа к полям опущены для экономии места). Данные классы представлены в листинге 2.

Листинг 2 – Описание классов User и Role

```
class User {  
    private String id;  
    private String name;  
    private List<Role> roles;  
}  
  
class Role {  
    private String id;  
    private String name;  
}
```

Теперь для данной простой иерархии классов введем так называемые маппинги (отображения), представленные в листинге 3.

Листинг 3 – Пример маппинга на доменные классы

```
<resultMap type="User" id="userMap">  
    <result id="id" column="u_id"/>  
    <result property="name" column="u_name"/>  
    <collection property="roles" ofType="Role" resultMap="roleMap"/>  
</resultMap>  
  
<resultMap type="Role" id="roleMap">  
    <result id="id" column="r_id"/>  
    <result property="name" column="r_name"/>  
</resultMap>
```

Как можно видеть из листинга, маппинги довольно просты.

Таким образом, мы задаем правило того, как должны будут заполняться наши объекты из выборки данных. MyBatis будет брать значение колонки и вставлять его в объект при помощи соответствующего setter-а (метода установки значения).

В данном примере используется коллекция ролей. Однако если есть необходимость реализовать ассоциацию (один-к-одному связь), то в таком случае в маппере будет использоваться **association** тег.

Идентификаторы мапперов должны быть уникальными (аттрибут *id* в теге *resultMap*).

Для наглядности на рисунке 1 представлена схема БД для данных сущностей.

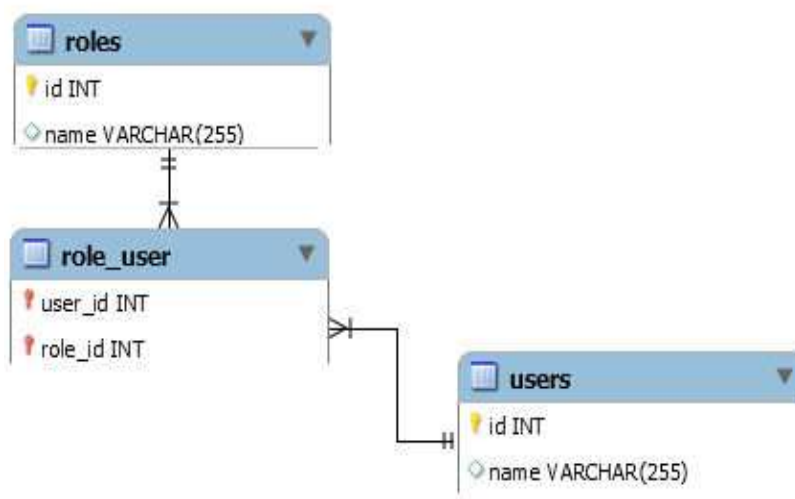


Рисунок 1. – Схема базы данных

Можно привести пример обычного запроса, который должен вернуть пользователя со всеми его ролями по его id (листинг 4).

Листинг 4 – Пример запроса выборки данных

```
<select id="getUserById" resultMap="userMap">
  SELECT
    u.id AS u_id,
    u.name AS u_name,
    r.id AS r_id,
    r.name AS r_name
  FROM users AS u
  INNER JOIN user_role AS ur
    ON u.id = ur.user_id
  INNER JOIN roles AS r
    ON ur.role_id = r.id
  WHERE u.id = #{id}
</select>
```

Псевдонимы для выбираемых колонок должны полностью совпадать со значениями атрибутов *column* в мапперах. В данном примере используется элемент в фигурных скобках “#{id}” – это тот параметр, который мы передаем в данный запрос. Как это вызывается из Java-кода можно увидеть в листинге 5.

Листинг 5 – Пример вызова запроса из Java-кода

```
class User {
  private SqlSessionTemplate template;
  public User getUserById(String id) {
    return template.selectOne("IUserMapper.getUserById", id);
  }
}
```

SqlSessionTemplate является тем классом, через который происходит обращение к БД. В примере выше мы выбираем один элемент по id, который принимаем в качестве параметра метода.

Метод *selectOne* первым параметром принимает имя маппера и id запроса, который требуется выполнить, а вторым параметром уже сам параметр запроса.

Параметры, передаваемые в запросы, могут быть не только примитивными типами, но и сложными пользовательскими объектами. В таком случае для обращения к полям этих параметров следует обращаться через точку. Например, для параметра типа *User*, переданного в запрос, можем обратиться к его имени так: *#{user.name}*.

Для выполнения вставки, удаления, обновления данных используются теги *insert*, *delete*, *update*, соответственно. Кроме того, имеется возможность переиспользовать части SQL запросов, как представлено в листинге 6.

Листинг 6 – Пример переиспользования фрагмента SQL-запроса

```
<sql id="filtered">
  FROM users AS u
  INNER JOIN user_role AS ur
    ON u.id = ur.user_id
  INNER JOIN roles AS r
    ON ur.role_id = r.id
  WHERE u.id = #{id}
</sql>
-----
<select id="getUserById" resultMap="userMap">
  SELECT
    u.id AS u_id,
    u.name AS u_name,
    r.id AS r_id,
    r.name AS r_name
  <include refid="filtered"/>
</select>
```

Когда один и тот же фрагмент SQL-запроса используется в нескольких местах, уместно будет вынести этот фрагмент, как показано выше, и ссылаться на него в остальных местах. Это экономит место, а также способствует уменьшению объема работы, если данный фрагмент запроса в какой-то момент необходимо изменить. При таком подходе изменения будут вноситься только в одном месте, что экономит еще и время на внесение изменений.

Заключение

Представлен краткий обзор базовых возможностей MyBatis фреймворка и его базовой конфигурации. Также представлены примеры маппингов и запросов к БД.

Приведен пример вызова запросов из Java-кода.

Как видно, MyBatis не упраздняет наличие SQL-кода в проекте, однако он делает его написание и сопровождение гораздо легче.

Данная работа может служить отправной точкой в изучении фреймворка MyBatis.

ЛИТЕРАТУРА

1. Хабрахабр [Электронный ресурс] MyBatis как более быстрая альтернатива Hibernate. – Режим доступа: <https://habrahabr.ru/post/247885>. – Дата доступа: 27.09.2017.
2. Tutorialspoint [Электронный ресурс] MYBATIS Tutorial. – Режим доступа: <http://www.tutorialspoint.com/mybatis/>. – Дата доступа: 27.09.2017.
3. MyBatis [Электронный ресурс] Introduction. – Режим доступа: <http://www.mybatis.org/mybatis-3/>. – Дата доступа: 27.09.2017.

УДК 004.03

**СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИСПОЛЬЗОВАНИЯ ОБЛАЧНЫХ РЕШЕНИЙ
И СОБСТВЕННЫХ СЕРВЕРОВ ПРИ РАЗРАБОТКЕ ВЕБ ПРИЛОЖЕНИЙ****А.И. СТАТУТ***(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются плюсы и минусы двух подходов при разработке веб приложений. Приведены примеры технологий для разработки веб приложения в инфраструктуре Интернет-компании Amazon. Рассмотренные облачные технологии позволяют с легкостью развернуть и в будущем масштабировать веб приложения, однако обладают повышенной стоимостью и сложностью при миграции к другой компании либо на свой выделенный сервер.

Большое количество хостингов, которые доступны современным разработчикам и предприятиям, делает сложным первоначальный выбор, где разместить свое веб приложение. Выбор осложняет также различные типы решений, таких как выделенный сервер либо, так называемое облако, и каждый из этих типов предоставляет определенные преимущества. Важно, чтобы индивидуальные разработчики и компании подбирали решения для веб приложений исходя из своих потребностей и сильных сторон платформы хостинга.

Облачные вычисления

Облачный хостинг – это платформа виртуализированных серверов, где каждый облачный сервер является виртуальной машиной, которая работает на мощном выделенном сервере, и выделенный сервер поддерживает несколько облачных серверов [1]. Поскольку каждый виртуальный сервер является программной конструкцией, администратор может контролировать его, как и любое другое программное обеспечение. Виртуальный сервер можно создать за считанные секунды и при необходимости он может предоставить больше ресурсов, чем запрашивалось первоначально. Также виртуальный сервер может быть немедленно уничтожен, когда он больше не нужен.

Стоит также отметить, что облачные сервера могут быть настроены таким образом, чтобы обеспечить необходимые уровни производительности, безопасности и управления, аналогичные тем, какие предоставляются выделенными серверами, но вместо того, чтобы размещаться на физическом оборудовании, предназначенном исключительно для вас, они находятся в общей «виртуализированной» среде, которой управляет ваш провайдер облачных серверов. Благодаря этому, вы получаете выгоду от эффекта масштаба обмена оборудованием с другими клиентами. Поэтому облачные решения выгодны для малых и средних предприятий для оптимизации своих затрат, связанных с приобретением и управлением выделенной инфраструктурой. С облачными серверами клиент платит только за точный размер серверного пространства, которое он использует. Также присутствует возможность легкого масштабирования ресурсов, как и увеличение их, когда потребности клиента растут, так и уменьшения, чтобы избежать дополнительной платы за простой инфраструктуры, когда нагрузка минимальна.

Выделенные сервера

Выделенный сервер – это физический сервер, который приобретается либо сдается в аренду. Выделенные сервера отлично подходят для крупных предприятий и организациям, которым требуется исключительно высокий уровень безопасности данных. В отличие от облачных решений, все ресурсы выделенного сервера доступны клиенту.

Выделенный сервер может быть настроен в соответствии с точными требованиями клиента, который может выбрать процессор, оперативную память и хранилище, которые наилучшим образом соответствуют его потребностям. Выделенный сервер – это самый мощный вариант среди односерверного хостинга и единственным более мощным вариантом, чем выделенный сервер, является кластер выделенных серверов. Предприятия, которые используют приложения с большим количеством операций чтения и записей, например, базы данных либо обработчики большого количества данных, находят огромные преимущества в выделенных серверах.

Основные различия в облачных решениях и выделенных серверах

Если клиенту нужно разместить свое веб приложение в сети Интернет, то с помощью облачных серверов это можно сделать в считанные минуты. Если размещать веб приложение на выделенных серверах, то их покупка и настройка может занять несколько дней.

Если клиенту необходимо изменить размер выделенных ресурсов, таких как размер оперативной памяти, размер дискового пространства либо изменить процессор, то с облачными серверами это сделать можно также гораздо быстрее и в автоматическом режиме в отличие от выделенных серверов, где эта операция требует ручного процесса обновления.

Также стоит отметить, что автоматическое масштабирование доступно также только среди компаний, предоставляющих облачный хостинг.

Однако, если предприятию необходим полный контроль над своими приложениями и серверами, а также строгая безопасность и соответствие требованиям, то выделенные сервера будут лучшим выбором.

Не стоит забывать и про соотношения цены и производительности, в этом лучшее преимущество также за выделенными серверами.

Пример технологий для развертывания веб приложения в облачной платформе Amazon

Рассмотрим несколько технологий, с помощью которых можно легко управлять и масштабировать веб-приложения в облачной платформе компании Amazon [2].

AWSLambda – вычислительный сервис, который позволяет запускать код приложения без создания или настройки собственных серверов. AWSLambda выполняет код только в случае необходимости и автоматически масштабируется с нескольких запросов в день до тысяч в секунду. Клиент платит только за время вычислений, которые приложение непосредственно использует.

AmazonDynamoDB – это полностью управляемый сервис NoSQLбазы данных, обеспечивающий быструю и прогнозируемую производительность с плавной масштабируемостью. Клиент может использовать DynamoDB для создания таблиц базы данных, которые могут хранить и извлекать любой объем данных и обслуживать любой уровень трафика запросов. DynamoDB также автоматически распределяет данные и трафик для таблицы на достаточное количество серверов для обработки заданной клиентом емкости запроса и объема данных, сохраняя при этом постоянную и быструю работу.

AmazonS3 – это хранилище для статических данных, которое может быть использовано для хранения и получения любого количества данных в любое время.

Помимо вышеперечисленных технологий, Amazon предлагает большое количество облачных решений для нужд любого веб приложения.

Заключение

В действительности, облачные решения и выделенные сервера поддерживают любой тип приложений и сервисов, но каждый из них лучше всего подходит для конкретных сценариев и выбор правильной платформы может помочь компаниям снизить затраты, повысить гибкость и максимально использовать инфраструктуру за приемлемые деньги. Рассмотренные облачные технологии компании Amazon позволяют с легкостью развернуть и в будущем масштабировать веб приложения, однако обладают недостатками, такими как повышенная стоимость и сложность при миграции к другой компании либо на свой выделенный сервер.

ЛИТЕРАТУРА

1. Cloud computing: state-of-the-art and research challenges [Электронный ресурс]. – Режим доступа: <https://link.springer.com/content/pdf/10.1007%2Fs13174-010-0007-6.pdf>. – Дата доступа: 15.09.2017.
2. Документация AWS [Электронный ресурс]. – Режим доступа: <https://aws.amazon.com/documentation/>. – Дата доступа: 18.08.2017.

УДК 004.03

**СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ
И СРЕДСТВ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ****А.И. СТАТУТ***(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются методы разработки мобильных приложений, такие как разработка нативных приложений и гибридных. Также в статье произведен анализ преимуществ и недостатков разработки мобильных приложений с помощью различных типов программных библиотек.

Существует множество различных библиотек и других средств и инструментов разработки мобильных приложений. Каждый из них имеет свою особенность и предоставляет уникальные возможности, поэтому в данном исследовании поставлена задача – рассмотреть те инструменты, которые являются наиболее подходящими для реализации большинства мобильных приложений.

Операционные системы

Однако перед тем как говорить о видах мобильных приложений, стоит рассмотреть мобильные операционные системы (платформы).

На сегодняшний момент, по официальным данным, среди мобильных операционных систем в мире (и в частности в Беларуси) лидирует Android и iOS [1].

С учетом этих данных можно обосновать выбор операционных систем для разработки мобильного приложения, так как чем больше людей используют ту или иную платформу, тем выше вероятность того, что разрабатываемое приложение станет распространенным. Это утверждение не означает, что если разработано приложение под Android, то его сразу же загрузят тысячи людей, однако, если же приложение ориентированно на людей, проживающих в Республике Беларусь, и под операционную систему Windows Phone, то оно определенно обречено на провал, так как пользователей данной системы в нашей стране крайне мало.

Таким образом, при разработке приложения нужно учитывать предпочтения пользователей, на которых вы ориентируетесь. Для разработки мобильных приложений следует использовать наиболее распространенные операционные системы, какими в Республике Беларусь являются Android и iOS.

Типы мобильных приложений

Существуют два типа мобильных приложений: нативные и гибридные[2].

Нативные приложения разрабатываются инструментами, предоставленными Apple для iOS, Google для Android, Microsoft для Windows Phone и т.д., и стандартным языком для выбранной платформы, используется API, которое предоставляет данная платформа; интерфейс таких приложений организован и соблюдает стиль платформы.

Гибридные приложения чаще всего разрабатываются с помощью JavaScript, HTML5, CSS3. Инструменты, используемые для их написания, преобразуют веб-представление приложения в нативное, но в случае, если ведется разработка сложного приложения, уровни абстракции, используемые при преобразовании, чаще всего не позволяют использовать большинство, свойственных нативным приложениям, функциональных особенностей. Скорость работы таких приложений намного ниже нативных, а также они нуждаются в постоянном и непрерывном интернет-соединении. Перед тем как заняться разработкой, следует решить, для каких мобильных платформ будет реализовано приложение, сколько денежных средств будет вложено в проект.

Лучшим решением будет написание нативного приложения. Это обусловлено тем, что нативные приложения работают намного быстрее, а их интерфейс более стилизован. Но в этом случае затраты на разработчиков будут намного выше, так как потребуется команда специалистов с набором конкретных навыков. Если же требуется написать приложение, ориентированное на несколько платформ с наименьшими затратами и с удовлетворительным качеством, стоит задуматься о разработке гибридного приложения.

Анализ гибридных платформ

Если требуется разработать гибридное приложения, то, учитывая все недостатки данного вида приложений, необходимо отобрать несколько наиболее распространенных средств разработки и провести их сравнительный анализ. Рассмотрим следующие средства разработки:

1. IONIC.
2. AppceleratorTitanium.
3. ReactNative.
4. Cordova.

IONIC – это средство разработки приложений с открытым кодом для разработки мобильных гибридных приложений. Построен на основе AngularJS и Apache Cordova и обеспечивает разработку, используя такие веб-технологии как CSS, HTML5 и JavaScript. Приложения, созданные с помощью этих технологий распространяются через магазины приложений (Google Play или App Store и т.д.) [3].

Кроме функций, унаследованных от Apache, Cordova также предоставляет следующие возможности:

1. Выбор шаблона приложения.
2. Сборка и запуск в эмуляторе, на реальном устройстве, в браузере.
3. Генерация иконок и других элементов.

Поддерживаемые платформы: Android, iOS, Windows, BlackBerry.

Appcelerator Titanium – это фреймворк с открытым кодом, который позволяет создавать гибридные мобильные приложения с помощью JavaScript, разработанный Appcelerator.

Архитектурой этой библиотеки является кроссплатформенное API для доступа к нативным компонентам пользовательского интерфейса таким как: панели навигации, меню, диалоговым окнам, файловой система, сети, геолокации, акселерометр и карты. Весь исходный код приложения интерпретируется с использованием механизма JavaScript. Загрузка программы занимает больше времени, чем нативная программа, так как интерпретатор и все необходимые библиотеки должны быть загружены до интерпретации исходного кода. Некоторые разработчики сообщают: хотя работа с титаном дает быстрые результаты, что хорошо подходит при создании прототипов, однако есть вопросы, связанные с различиями в поведении с кроссплатформенным API, стабильностью и управлением памяти, что заставляло их переписывать свои приложения с помощью нативного кода.

Поддерживаемые платформы: Android, iOS, Windows, BlackBerry.

React Native – это платформа для создания гибридных приложений на основе JavaScript библиотеки React [4]. В отличие от других подобных платформ, которые запускают JavaScript приложения в веб-браузере, React Native компилирует в машинный код для соответствующей платформы. Это означает, что никакого снижения производительности приложения, созданного с помощью него не будет и приложение будет работать также как, как и приложения построенные с помощью нативных инструментов. Однако, кроме очевидных плюсов, платформа имеет множество недостатков: старые версии платформы не поддерживаются новыми, нет хорошей документации, всю необходимую информацию придется искать на форумах, трудоемкий процесс отладки приложений. Их можно объяснить тем, что платформа вышло относительно недавно и очень стремительно развивается.

Поддерживаемые платформы: Android, iOS.

Apache Cordova – одна из наиболее популярных платформ для разработки мобильных приложений. Она позволяет создавать гибридные мобильные приложения для мобильных устройств с помощью CSS3, HTML5, JavaScript. Приложения выполняются в обертках, которые являются платформо-ориентированными и опираются на стандартные API привязки для доступа к возможностям каждого из устройств, таких как датчики, данные, состояния сети и т.д. [5].

Поддерживаемые платформы: Apple iOS, Bada, BlackBerry, FirefoxOS, GoogleAndroid, LGwebOS, MicrosoftWindowsPhone, Tizen (SDK 2.x) и UbuntuTouch.

Если сравнить приведенные выше платформы, то среди них мы можем выделить IONIC и React Native, так как они являются наиболее быстрыми и продвинутыми по сравнению с Apache Cordova и Appcelerator Titanium. Ниже рассмотрим их сравнительный анализ на основе простейшего приложения с картой.

Сравнение платформы IONIC и ReactNative

Рассмотрим сравнение простейшего приложения, содержащего карту, разработанного на IONIC и React Native. Сравнение будет производиться по следующим критериям: сторонние модули, размер приложения, использование памяти.

1. Сторонние модули

Разрабатывая на React Native, мы можем использовать сторонние модули. В данном случае будем использовать нативное представление Google карт. В IONIC мы также можем использовать сторонние плагины, однако в случае с картами мы должны использовать WebView для их отображения. Недостатком данного подхода является более долгая загрузка и использование большего объема памяти. В нативном представлении карты мы можем использовать трехмерное отображение, масштабирование и компас.

2. Размер приложения

IONIC использует уже существующее WebView для отображения приложения, поэтому его размер достаточно небольшой. В то же время React Native создает мост между JavaScript и нативными компонентами, что сказывается на размере приложения. Одно и то же приложение в IONIC имеет размер всего 2.88 MB, в то время как в React Native – 8.22 MB. Разница довольно велика, хотя файлы приложения React Native можно сжать с помощью ProGuard. Однако в данном случае это сжатия не было выполнено.

3. Использование памяти

Количество памяти, которое использует приложение, очень важно, особенно для медленных устройств на Android. Поэтому там необходимо сохранить низкий уровень использования памяти так, чтобы приложение могло бесперебойно работать на устройствах. Если мы запустим команду `adb shell dumpsys meminfo`, то получим следующий результат: Ionic Framework использует ~ 84,714 kB памяти, в то время как React Native – ~ 58,585 kB. Таким образом, приложение работает лучше на React Native.

Подводя итог можно сказать, что наилучшим средством разработки гибридного мобильного приложения является React Native.

Заключение

Подводя итог, можно сказать, что выбор разрабатывать нативное либо гибридное мобильное приложение необходимо тщательно проанализировать, так как у каждого подхода есть свои преимущества и недостатки.

Если необходимо разработать приложение в сжатые сроки и на несколько операционных систем одновременно, но, возможно, в ущерб производительности, то лучше выбрать гибридные платформы.

Если проект имеет достаточный бюджет, приложение будет иметь сложные элементы интерфейса и важна производительность, то выбор за нативным методом.

Если сравнивать гибридные платформы, ReactNative является одним из лучших решений как компромисс между другими гибридными платформами и нативным способом разработки.

ЛИТЕРАТУРА

1. Восемь факторов, влияющих на стоимость разработки мобильного приложения [Электронный ресурс]. – Режим доступа: <https://link.springer.com/content/pdf/10.1007%2Fs13174-010-0007-6.pdf>. – Дата доступа: 10.09.2017.
2. Мобильные приложения: нативные vs html5 vs гибридные [Электронный ресурс]. – Режим доступа: <http://www.skywell.com.ua/blog/mobilnyie-prilozheniya-nativnyie-vs-html5-vs-gibridnyie/>. – Дата доступа: 07.08.2017.
3. Ionic framework. Обзор экосистемы [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/company/simpleweek/blog/254681/>. – Дата доступа: 07.08.2017.
4. React Native [Электронный ресурс]. – Режим доступа: <https://facebook.github.io/react-native/>. – Дата доступа: 07.09.2017.
5. Введение в Apache Cordova [Электронный ресурс]. – Режим доступа: <https://cordova.apache.org/docs/ru/latest/guide/overview/>. – Дата доступа: 17.09.2017.

УДК 004.021

**WEB-SCRAPING КАК ИНСТРУМЕНТ ИЗВЛЕЧЕНИЯ
ПОЛЕЗНЫХ ДАННЫХ ИЗ СЕТИ ИНТЕРНЕТ****Н.О. ШЕРШНЁВ****(Представлено: Ю.Н. КРАВЧЕНКО)**

Рассматриваются преимущества и недостатки получения данных из сети Интернет способом Web-scraping. Показаны основные способы анализа и обработки данных, их назначение и применение в повседневной жизни человека.

Информация играет чрезвычайно важную роль в жизни человека. Известно, что «Кто владеет информацией, тот владеет миром». В процессе развития современного общества все большие объемы несортированных данных хранятся в сети Интернет в открытом доступе. Поэтому при создании нового веб-ресурса перед разработчиком стоит выбор: использовать готовые данные из сети Интернет или прикладывать усилия для создания собственного контента. Это обусловило создание и разработку технологий, которые бы позволяли собирать полезную информацию в интернете, анализировать ее и предоставлять в структурированном виде конечному пользователю.

Веб-скрапинг называется такой подход к извлечению полезных данных веб-скрапингом, который подразумевает под собой создание специального программного обеспечения, позволяющего получать пользователю всю необходимую информацию с одного или нескольких интернет-ресурсов.

Существующие веб-скраперы имеют узконаправленную специализацию и зачастую создаются для конкретного веб-ресурса, что предполагает большие человеческие усилия для автоматизации процессов получения и дальнейшего преобразования информации к структурированному виду.

Веб-скраперы имеют широкое применение в разных сферах жизни человека.

При помощи веб-скраперов можно решить следующие задачи:

- мониторинг данных о погоде;
- сбор личных данных пользователей;
- поиск вакансий;
- работа с частными объявлениями по поиску жилья;
- отслеживание цен на товары в различных магазинах и т.д.

Среди готовых решений для скрапинга веб-сайтов стоит отметить следующие:

- веб-сервисы, которые работают через API (DiffBot, Embedly и др.);
- проекты с открытым кодом (Goose, Goutte, Morph, Scrapy и др.) [1].

Далее рассмотрим основные способы анализа и извлечения данных, использующие технологию веб-скрапинга:

1. **Copy-and-paste.** В некоторых случаях лучшим решением для получения данных из сети Интернет является простое копирование необходимой информации пользователем. Также этот способ полезен в том случае, когда веб-скрапер не может преодолеть защиту веб-сайта от машинной автоматизации.

2. **Text pattern matching.** Довольно простой, однако мощный подход к извлечению информации с веб-сервиса может быть основан на использовании регулярных выражений.

3. **HTTP programming.** Данный способ основывается на отправке http-запросов к удаленному веб-серверу, используя программирование сокетов.

4. **DOM parsing.** Программа, созданная на основе данного подхода, встраивается в полноценный веб-браузер, например Internet Explorer или Mozilla Firefox. Система управления браузером анализирует веб-страницы в DOM-дереве, на основе которых программа получает необходимую информацию.

5. **Computer vision web-page analysis.** Существуют попытки создания такого программного обеспечения, которое использует машинное обучение и компьютерное зрение, идентифицирует и извлекает информацию с веб-страниц.

6. **HTML parsing.** Многие веб-сайты содержат коллекции страниц, которые были сгенерированы автоматически из таких источников, как базы данных. Данные обычно кодируются в похожие страницы с помощью общих скриптов и шаблонов. В процессе интеллектуального анализа данных, программу, которая определяет такие шаблоны, извлекает их содержимое и переводит в понятную форму, называют оболочкой или wrapper [2].

7. **Web-scraping.** Существует программное обеспечение, которое может быть использовано для настройки веб-скрапинг решений. Такое программное обеспечение может автоматически распознать структуру веб-страницы, что освобождает от написания веб-скрапинг кода [2].

Заключение

В ходе проведенного исследования рассмотрены основные способы анализа и извлечения данных, использующие технологию веб-скрапинга:

- Web-scraping как инструмент извлечения полезных данных из сети Интернет;
- основные способы анализа и обработки данных;
- назначение и применение их в повседневной жизни человека.

ЛИТЕРАТУРА

1. Википедия [Электронный ресурс] Web-scraping. – Режим доступа: https://en.wikipedia.org/wiki/Web_scraping. – Дата доступа: 26.09.2017.
2. Habrahabr [Электронный ресурс] Web-scraping с помощью Python. – Режим доступа: <https://habrahabr.ru/post/280238/>. – Дата доступа: 26.09.2017.
3. Habrahabr [Электронный ресурс] Web-scraping при помощи Node.js. – Режим доступа: <https://habrahabr.ru/post/301426/>. – Дата доступа: 26.09.2017.

УДК 004.021

**МОБИЛЬНОЕ ПРИЛОЖЕНИЕ «МУЗЫКАНТЫ РОССИИ»
ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ АНДРОИД****Н.О. ШЕРШНЁВ****(Представлено: Ю.Н. КРАВЧЕНКО)**

Рассматриваются назначение и функциональные возможности разработанного автором мобильного приложения «Музыканты России» для операционной системы Андроид. Приложение бесплатное и распространяется посредством официального магазина приложений Google Play Market. В настоящее время идет процесс развития данного проекта в Беларуси и Казахстане.

Музыка является неотъемлемой частью жизни современного общества. Миллионы людей по всему миру увлекаются музыкой, для многих это стало смыслом жизни. Постоянно развивающаяся музыкальная индустрия обуславливает появление новых музыкантов и музыкальных групп. Однако путь к известности тернист, и не каждый начинающий добивается внимания к своему творчеству. Поэтому помощь в развитии и становлении юных дарований очень важна. На помощь таким людям приходит проект «Музыканты России», который был создан в марте 2015 года.

Описание проекта

Данный проект предназначен для облегчения коммуникации между музыкантами и участниками музыкальной индустрии, обмена полезной информацией, поиска единомышленников для совместного творчества. Проект является уникальным на территории СНГ, поэтому он нашел поддержку у множества спонсоров.

Благодаря проекту «Музыканты России» налажено сотрудничество со многими музыкальными партнерами по всей стране. Под музыкальными партнерами понимаются: концертные площадки и репетиционные базы, музыкальные лейблы и интернет-магазины, учебные базы, а также концертные агентства. Разработана собственная программа лояльности, которая подразумевает предоставление клиентам персональных бонусных карт со скидками на услуги музыкальных партнеров. Вся эта полезная информация предоставлялась пользователям посредством веб-сервиса. Идея зарекомендовала себя хорошо и получила множество положительных отзывов от пользователей, однако для развития проекта дальше было принято решение о создании мобильного приложения «Музыканты России» для операционной системы Андроид.

Первая версия приложения была запущена в августе 2016 года. На данном слайде представлен первоначальный дизайн приложения. Для улучшения качества предоставляемых услуг в марте 2017 года было принято решение о внесении изменений в приложение, добавление нового функционала, а также создание нового дизайна.

Мобильное приложение разрабатывалось для операционной системы Андроид, на языке программирования Java. Для удобства навигации по музыкальным партнерам использовались карты Google Maps. Серверная часть разрабатывалась при помощи PHP и MySQL. Все выбранные технологии использовались по требованию заказчика.

Мобильное приложение «Музыканты России» обладает следующими функциональными особенностями:

- возможность выполнять работу с информацией, поступающей в разрабатываемое приложение, что включает в себя обеспечение пользователя удобными средствами ввода, а также возможность редактирования, добавления и удаления информации;
- наличие визуальной среды для просмотра и обработки информации, важными характеристиками которых являются дизайн пользовательского интерфейса, интуитивность и удобство навигации;
- возможность получения информации о музыкальных магазинах, репетиционных базах и концертных площадках. Предоставление этой информации в виде списков и при помощи карты Google Map;
- возможность получение последней актуальной информации о предстоящих мероприятиях в мире музыки;
- возможность выставления оценки для музыкального партнера;
- возможность редактирования личной информации пользователя;
- возможность подачи частного объявления о поиске музыкантов, или музыкальных коллективов для дальнейшего сотрудничества.

Заключение

Рассмотрено мобильное приложение «Музыканты России» для операционной системы Андроид, его назначение и функциональные возможности.

Приложение бесплатное и распространяется посредством официального магазина приложений Google Play Market.

Общее количество скачиваний приложения составило 7 096, за последние два месяца приложение скачали 2 728 раз.

Приложение в дальнейшем планируется развивать. В настоящее время идет процесс развития данного проекта в Беларуси и Казахстане.

ЛИТЕРАТУРА

1. Википедия [Электронный ресурс] Web-scraping. – Режим доступа: https://en.wikipedia.org/wiki/Web_scraping. – Дата доступа: 26.09.2017.
2. Habrahabr [Электронный ресурс] Web-scraping с помощью Python. – Режим доступа: <https://habrahabr.ru/post/280238/>. – Дата доступа: 26.09.2017.
3. Habrahabr [Электронный ресурс] Web-scraping при помощи Node.js. – Режим доступа: <https://habrahabr.ru/post/301426/>. – Дата доступа: 26.09.2017.

УДК 004.77

ПОНЯТИЕ ТЕХНОЛОГИИ БЛОКЧЕЙН

К.С. ГОЛОВАН

(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Раскрываются понятия блокчейна, децентрализованной сети, разрешения консенсуса, частного и публичного блокчейна. Приводятся примеры современных блокчейн-решений. Блокчейн позволяет безопасно распространять и/или обрабатывать данные между несколькими лицами через недоверенную сеть. Наиболее интересным вариантом данных является возможность передачи информации, которая требует наличия третьей доверенной стороны.

Блокчейн – это журнал с фактами, реплицируемый на несколько компьютеров, объединенных в сеть равноправных узлов (P2P). Фактами может быть что угодно – от денежных операций и до подписания контента. Члены сети – анонимные лица, называемые узлами. Все коммуникации внутри сети используют криптографию, чтобы надежно идентифицировать отправителя и получателя. Когда узел стремится добавить факт в журнал, в сети формируется консенсус, чтобы определить, где этот факт должен появиться в журнале; этот консенсус называется блоком.

Децентрализованные сети с равноправными узлами появились задолго до блокчейна. К примеру, BitTorrent, только вместо обмена файлами в блокчейне обмениваются фактами.

P2P-сетям, как и прочим распределенным системам, приходится решать сложную проблему информатики – разрешение конфликтов, или согласование. Реляционные базы данных предлагают ссылочную целостность, но такой особенности нет в распределенной системе. Если два несовместимых факта прибывают в одно и то же время, система должна иметь правила для определения того, какой факт считать правильным.

В P2P сетях, два факта отправленные примерно в одно время могут прибыть в разном порядке в удаленные узлы. Тогда, каким образом всей сети согласовать, какой факт пришел первым?

Чтобы гарантировать целостность в P2P сети, вам нужен способ согласования порядка фактов. Вам нужна система консенсуса.

Алгоритмы консенсуса для распределенных систем это очень активное поле для исследований [1].

Задача распределенного консенсуса не специфична для блокчейнов и имеет хорошо проверенные решения для многих других распределенных систем (например, баз данных NoSQL). Но биткойн и другие блокчейны от предыдущих наработок отличаются условиями работы сети.

В обычных алгоритмах византийского консенсуса у узлов сети есть «личности», выражаемые через цифровые подписи или сходные криптопримитивы, а сам список узлов известен заранее или меняется редко, но предсказуемо. В биткойн-блокчейне все наоборот.

Участники сети не только заранее неизвестны, но и могут свободно подключаться или отключаться от сети. При этом блокчейн, являясь децентрализованной системой, имеет определенные свойства: устойчивость к цензуре (никто не может запретить майнить криптовалюту) и объективность (для определения текущей версии журнала транзакций не нужно доверие неким авторитетным источникам – корень доверия находится в самом блокчейне).

Из-за этого обычные алгоритмы византийского консенсуса для блокчейна не подходят. Поэтому было предложено множество различных алгоритмов, среди которых выделяются две основные категории: алгоритмы на основе доказательства работы (proof-of-work) и алгоритмы на основе подтверждения доли (proof-of-stake).

В случае доказательства работы хеш сообщения, объединенного со специальным полем (nonce), должен быть меньше определенного значения (или начинаться с определенного числа нулевых битов). Nonce не несет смысла для самого сообщения – это поле перебирается автором доказательства, пока не будет найдено подходящее значение. Название «доказательство работы» отражает тот факт, что для нахождения nonce надо совершить вычислительную работу, ожидаемое количество которой измеримо. Например, если нужно, чтобы первые 16 бит хеша равнялись нулю, в среднем нужно перебрать 65 536 значений nonce.

Доказательство работы похоже на цифровые подписи – оно обеспечивает целостность сообщения, так как вероятность того, что один и тот же nonce подойдет для различных сообщений, очень мала. Доказательства также легко проверяются – достаточно провести лишь одну операцию хеширования. В отличие от подписей, создание доказательства работы не требует знания секретов, но «потребляет» больше вычислительных ресурсов. Доказательства работы используются узлами биткойна для определения состояния системы. Актуальный журнал транзакций определяется как цепочка блоков с наибольшей сум-

марной сложностью доказательств работы. Майнеры, соответственно, должны искать блок поверх этой цепочки. Но, теоретически, никто не запрещает создавать новые блоки на основе какого-то старого блока (иногда случаются расщепления – форки – блокчейна, потому что два майнера находят новый блок практически одновременно). Однако намеренные форки невыгодны экономически, потому что блоки из побочных ветвей блокчейна никем не учитываются и не приносят их создателям никакой прибыли – одни затраты на нахождение доказательства работы.

Консенсус Накамото обеспечивает два ключевых требования к блокчейну, которые мы выделили ранее. Из-за того, что доказательство работы не привязывается к личностям майнеров (в отличие от цифровых подписей/сертификатов), обеспечивается устойчивость к цензуре. А за счет того, что доказательства работы быстро проверяются и не требуют для проверки ничего, кроме блокчейна, достигается эффективность протокола.

Консенсус Накамото устойчив к атакам на сеть блокчейна, причем его безопасность достаточно хорошо исследована теоретически (в отличие от более новых предложений для блокчейнов). В оригинальной статье отмечено, что сеть биткойна продолжит адекватно функционировать, даже если половина майнеров начнет злонамеренную активность. Если же среди майнеров появится злонамеренное большинство (так называемая атака 51%), они смогут игнорировать блоки от всех остальных майнеров, чтобы забрать себе всю награду в сети, или, например, мошенничать, производя повторную трату средств за счет намеренных форков блокчейна. Однако при этом отметим, даже если все майнеры в сети сговорятся, они не смогут обойти базовые механизмы безопасности биткойна – например, не смогут похитить биткойны пользователей.

У консенсуса Накамото есть свойство, которое многими воспринимается как большой недостаток – для обеспечения безопасности нужно «работать», то есть производить доказательства работы. Вычисления, которые выполняются в рамках PoW, не служат никакой полезной цели, и это архитектурная особенность. Оказывается, очень сложно придумать доказательство работы, которое бы выполняло общественно полезную роль. Поэтому может показаться, что ресурсы (прежде всего, электричество) на майнинг тратятся впустую (если не считать, что они тратятся на обеспечение безопасности).

Помимо этого, майнинг все же подвержен проблеме централизации. Более 70% хешрейта биткойна на данный момент находится в одной стране – Китае. Многие криптовалюты пытаются децентрализовать майнинг за счет использования доказательств работы, которые экономически невыгодно вычислять на специализированном оборудовании, однако с таким подходом возникает другая проблема – если сделать выгодным майнинг биткойнов с помощью домашних компьютеров, то арендовать оборудование для атаки (или создать для нее ботнет) станет существенно дешевле и проще.

Пытаясь решить эти проблемы, сообщество предлагает множество алгоритмов консенсуса, которые не требуют «работы». Самая популярная категория таких алгоритмов основана на доказательствах доли (proof-of-stake, PoS). Доказательство доли похоже на доказательство работы, только вместо совершения определенной работы автор показывает, что у него есть доля в системе (например, в виде ненулевого баланса криптовалюты).

Однако у доказательства доли есть досадный недостаток в сравнении с PoW: поскольку доказательства доли базируются не на реальном мире (вычислительная мощность), а на чем-то внутри самого блокчейна (баланс криптовалюты), задача построения надежного PoS-алгоритма оказывается нетривиальной [3].

Согласно последним исследованиям в области блокчейнов компании BitFury, выделяются следующие типы блокчейнов:

Открытый блокчейн (англ. publicblockchain) – блокчейн, в котором не существует ограничений на чтение данных блоков (при этом данные могут быть зашифрованы) и ограничений на отсылку транзакций для включения в блокчейн.

Закрытый блокчейн (англ. privateblockchain) – блокчейн, в котором прямой доступ к данным и отправка транзакций ограничен определенным узким кругом организаций.

Общедоступный (инклюзивный) блокчейн (англ. permissionlessblockchain) – блокчейн, в котором не существует ограничений на личность обработчиков транзакций (т.е. пользователей, которые могут создавать блоки транзакций).

Эксклюзивный блокчейн (англ. permissionedblockchain) – блокчейн, в котором обработка транзакций осуществляется определенным списком субъектов с установленными личностями.

Приватные блокчейны имеют определенные преимущества: *во-первых*, низкая стоимость транзакций, поскольку проверка их валидности проводится доверенными и высокопроизводительными узлами вместо десятков тысяч пользовательских устройств, как в случае с общедоступными сетями; *во-вторых*, блокчейн можно настроить таким образом, что показатель TPS (TPS – transactionspersecond) будет значительно большим, чем у общедоступных сетей (по крайней мере, в ближайшем будущем). Единственным ограничением в этом случае остается пропускная способность самого слабого узла в сети.

Еще одним преимуществом частных блокчейнов может являться больший контроль над системой со стороны компании. Суть в том, что частный блокчейн позволяет, например, быстро обновлять функциональность. Поэтому он привлекателен для учреждений, работающих с реестрами и системами учета, поскольку формирует контролируемую и прогнозируемую среду, по сравнению с общедоступными блокчейнами (о которых речь пойдет ниже).

Более того, создание блоков в частном блокчейне зачастую не требует «доказательства работы» (proof-of-work). В качестве примера можно привести протокол создания блоков, используемый в BitShares. Имеется установленное число обработчиков транзакций N , каждый из которых обладает парой ключей – секретным и открытым. Создатели блоков известны и определяются по цифровой подписи в заголовке.

Операторы формируют блоки по очереди через заданные временные интервалы. Порядок создания блоков или фиксирован, или перемешивается после полного цикла (N блоков). Если оператор не сумел сформировать блок в отведенное ему время, то он пропускает раунд. Если такое поведение стало результатом злоумышленников, то ситуация расследуется. Таким образом, если обработчики транзакций являются единственными потребителями данных блокчейна, можно построить надежный протокол создания блоков (например, немного усложнив приведенный выше алгоритм), который не будет использовать доказательство работы.

И хотя частные блокчейны могут и не использовать доказательство работы, этот протокол все-таки может быть подключен для повышения уровня защищенности, упрощения аудита и, как результат, повышения контроля над системой для конечных пользователей. По сути, доказательство работы переводит доверие к блокчейну из субъективного (доверие к системе эквивалентно доверию к контролирующей его организации) к объективному (доверие к системе вытекает из математических законов и гарантированно высокой экономической стоимостью атаки на систему, которая не зависит от личности атакующего).

Публичные или общедоступные блокчейны могут быть прочитаны любым пользователем, каждый из которых имеет право формировать транзакции. При этом операции защищаются механизмами криптографической верификации, такими как доказательство выполнения работы или подтверждение доли (proof-of-stake).

«Контролирует» публичный блокчейн сразу все сообщество участников сети – разработчики, пользователи, поставщики услуг, майнеры – которые обеспечивают целостность сети и удобство работы в ней. Эффективность работы сети достигается с помощью обновлений протокола, предотвращающих вредные изменения. Именно поэтому система позволяет создавать децентрализованные приложения с минимумом затрат на техническое обслуживание.

Также они предоставляют способ защиты пользователей приложения от разработчиков, ограничивая возможности последних. В приложениях на публичном блокчейне разработчик не может сам по себе изменять код или данные.

Помимо этого, публичные блокчейны обладают сетевыми эффектами. Первыми пользователями приложений, построенных на публичном блокчейне, зачастую являются пользователи других приложений на том же блокчейне, которые узнали о них благодаря эффекту взаимодействия программ. Например, мобильный кошелек, работающий на публичном блокчейне, может добавить функцию для взаимодействия этого приложения с другими распределенными приложениями на этом же блокчейне, значительно расширив свою пользовательскую базу.

Стоит отметить, что публичные блокчейны позволяют разрешить проблему передачи «товаров». Например, если пользователь A хочет продать домен пользователю B , то они сталкиваются с определенными трудностями. Если A передаст домен первым, то рискует не получить деньги, в ином случае B рискует не получить «товар». Для решения этой проблемы применяются посредники, взимающие проценты за проведение транзакции.

Однако если блокчейн имеет систему доменных имен и валюту, расходы сокращаются до нуля с использованием смарт-контракта. Первый отправляет программе домен, а второй – деньги. Проблем не возникает, поскольку программа является доверенной, так как действует в публичном блокчейне [2].

Заключение

Блокчейн позволяет безопасно распространять и/или обрабатывать данные между несколькими лицами через недоверенную сеть. Данными может быть что угодно, но наиболее интересным вариантом данных является возможность передачи информации, которая требует наличия третьей доверенной стороны. Примерами такой информации являются деньги (требуют участия банка), права на собственность (требуют участия нотариуса), договор на заем и т.д. В сущности, блокчейн устраняет необходимость в участии третьего доверенного лица.

С технической точки зрения блокчейн является новшеством, которое опирается на три понятия: P2P сети, асимметричная криптография и распределенный консенсус, основанный на решении математической задачи.

Блокчейн можно рассматривать как слабоинхронизированную базу данных, реплицируемую столько же раз, сколько узлов в сети, или как суперкомпьютер, образованный комплексом всех CPU/GPU входящих в него узлов.

Одно из главных преимуществ блокчейнов по сравнению с другими моделями распределенных баз данных – интеграция обработки информации, слежения за корректностью и безопасностью в единый протокол, минимизирующий влияние человеческого фактора. Из-за юридических и технических причин учреждения, в которых задействованы финансовые системы учета или реестры, в среднесрочной перспективе могут быть заинтересованы в использовании блокчейнов с ограниченным доступом к обработке транзакций.

Что касается публичных блокчейнов, то их преимущества (в частности, их прозрачность и открытость базовых технологий и протоколов) могут привести к тому, что технология заменит многие функции традиционных финансовых институтов, изменив принципы работы финансовой системы

ЛИТЕРАТУРА

1. Объяснение блокчейна для веб-разработчиков [Электронный ресурс] / Ч. 1 : Теория. – Франция, 2016. – Режим доступа: <https://marmelab.com/blog/2016/04/28/blockchain-for-web-developers-the-theory.html>. – Дата доступа: 30.09.2017.
2. Различия, достоинства, недостатки: публичные и приватные блокчейны [Электронный ресурс]. – Россия, 2016. – Режим доступа: <https://habrahabr.ru/company/bitfury/blog/324458/>. – Дата доступа: 30.09.2017.
3. «Алгоритмы консенсуса»: Подтверждение доли и доказательство работы [Электронный ресурс]. – Россия, 2016. – Режим доступа: <https://habrahabr.ru/company/bitfury/blog/327468/>. – Дата доступа: 30.09.2017.

УДК 004.9

ПРОГРЕССИВНЫЕ ВЕБ-ПРИЛОЖЕНИЯ

К.С. ГОЛОВАН

(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Раскрываются понятия нативных приложений, прогрессивных веб приложений, манифеста, ServiceWorkers. Приводится сравнение нативных и прогрессивных приложений.

Интерес к разработке приложений для мобильных операционных систем в сообществе разработчиков стабильно растет, количество инструментов и подходов позволяющих создавать приложения становится все больше. Примерно с 2005 года веб-технологии перешли от использования статических страниц на динамические, т.е. генерируемые сервером (PHP, ASP.NET) и клиентской стороной (AJAX), а также пришли к понятию респонсивный дизайн. Однако с выходом первой линейки телефонов iPhone данные технологии отошли на второй план, так как нативные приложения предлагали более быструю работу приложения и удобный интерфейс для пользователя. В середине 2010 года улучшения в HTML5, CSS3 и JavaScript наравне с развитием браузерных движков сделали так называемые гибридные приложения настоящей альтернативой нативным приложениям. Гибридные приложения «воспроизводят» нативный интерфейс и требуют установки из специального хранилища приложений. Комбинация скриптов, стилей и разметки дает возможность создавать интерфейсы без использования технологии Flash. Также гибридные приложения не имеют доступа по URL, поддерживают богатый пользовательский интерфейс и имеют доступ к операционной системе. В 2015 году компанией Google был представлен один из новых подходов к разработке веб-приложений, а именно прогрессивные веб-приложения (ProgressiveWebApps, прогрессивные веб-приложения).

Прогрессивные веб-приложения используют преимущества новых технологий, чтобы принести пользователю все самое лучшее из мобильных сайтов и нативных приложений. Они надежные, быстрые и привлекательные. Они доступны из безопасного источника и загружаются независимо от состояния сети.

PWA – это название группы приложений, которые используют стек Web технологий (JS + HTML + CSS) и позволяют соединить простоту использования Web сайта со специфичными для нативных приложений операционной системы UX и техническими возможностями.

Основное предназначение PWA увеличить конверсию, количество пользователей и удобство использования веб-приложений на мобильных устройствах.

ProgressiveWebApps является логическим продолжением AcceleratedMobilePages. Таким образом, если вы ранее создавали AMP приложения, то вам однозначно стоит обновить свое приложение к нормам PWA приложений. Если вы до этого ничего не слышали о AMP, то это не станет для вас проблемой во время изучения PWA.

PWA приложению необходимо соответствовать следующим требованиям:

- быть прогрессивным – работать с каждым пользователем вне зависимости от окружения, используя метод постепенного улучшения как основной принцип работы;
- адаптивным – подстраиваться под любое устройство: десктоп, смартфоны, планшеты или что либо другое;
- независимым от соединения, используя ServiceWorker приложение должно работать в оффлайн режиме при прерывании или отсутствии соединения;
- выглядеть нативно – приложение должно соответствовать привычным для пользователя способам взаимодействия и навигации;
- самообновляемым – приложение должно контролировать процесс автоматического обновления посредством ServiceWorker API;
- безопасным – посредством использования HTTPS предотвращать перехват и подмену данных;
- определяемым – посредством W3C манифеста и регистрации через ServiceWorker приложение идентифицируется как «приложение» поисковыми системами;
- удерживающим – используя технические возможности, мотивируем пользователя еще раз использовать приложение, например, посредством push уведомлений;
- легким в установке – позволяет «сохранить» приложение на устройстве пользователя посредством добавления PWA приложения в список установленных приложений без использования магазина приложений;
- легким в использовании – для начала использования приложения достаточно открыть URL. Установка приложения не обязательна.

Для создания прогрессивных веб-приложений используются следующие основные технологии: манифест и ServiceWorkers [1].

Манифест веб-приложения предоставляет информацию о приложении (такую как имя, авторство, иконку и описание) в формате JSON-файла. Цель манифеста – установить веб-приложение на домашний экран устройства, предоставляя пользователю более быстрый доступ и больше возможностей.

Манифест веб-приложения внедряется в HTML-страницу, с помощью тега ссылки в заголовке документа:

```
<link rel="manifest" href="/manifest.webmanifest">
```

Пример манифеста:

```
{
  "name": "HackerWeb",
  "short_name": "HackerWeb",
  "start_url": ".",
  "display": "standalone",
  "background_color": "#fff",
  "description": "A simply readable Hacker News app.",
  "icons": [{
    "src": "images/touch/homescreen48.png",
    "sizes": "48x48",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen72.png",
    "sizes": "72x72",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen96.png",
    "sizes": "96x96",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen144.png",
    "sizes": "144x144",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen168.png",
    "sizes": "168x168",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen192.png",
    "sizes": "192x192",
    "type": "image/png"
  }],
  "related_applications": [{
    "platform": "web"
  }, {
    "platform": "play",
    "url": "https://play.google.com/store/apps/details?id=cheeaun.hackerweb"
  }]
}
```

Манифест типового прогрессивного веб-приложения содержит параметры, которые определяют ожидаемый цвет фона для веб-приложения; описание того, что делает приложение; предпочтительный для разработчика режим отображения приложения; массив объектов изображений, которые могут быть

использованы как иконки приложения в различных контекстах, например в разных ОС; удобочитаемое имя, ориентацию [2].

На рисунке 1 представлен пример различного отображения приложения для разработчика.

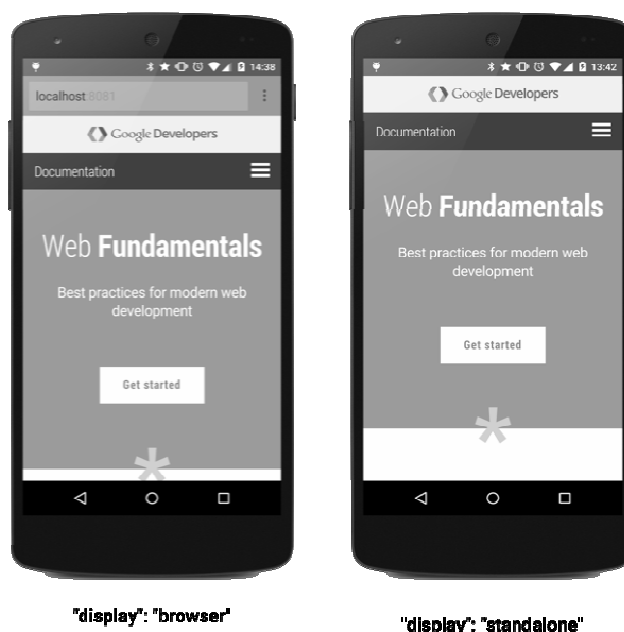


Рисунок 1. – Пример различного отображения прогрессивного веб-приложения при изменении параметра display в манифесте

Serviceworker – это событийно-управляемый worker, регистрируемый на уровне источника и пути. Он представляет собой JavaScript-файл, который может контролировать веб-страницу/сайт, с которым он ассоциируется, перехватывать и модифицировать запросы навигации и ресурсов, очень гибко кешировать ресурсы для того, чтобы предоставить вам полный контроль над тем, как приложение ведет себя в определенных ситуациях (например, когда сеть не доступна).

Serviceworker запускается в контексте worker'a, поэтому он не имеет доступа к DOM и работает в потоке отдельном от основного потока JavaScript, управляющего вашим приложением, а следовательно не блокирует его. Он призван быть полностью асинхронным; как следствие, синхронные API, такие как XHR и localStorage, в *serviceworker*'е использовать нельзя.

Из соображений безопасности *serviceworker*'ы работают только по HTTPS [3].

Serviceworkers также предназначены для использования для таких вещей, как:

- фоновая синхронизация данных;
- ответ на запросы от других источников;
- получение централизованного обновления для данных, сложных для расчетов, таких как геолокация или гироскоп, так что несколько станций может использовать одни и те же данные;
- компиляция на клиентской стороне и управление зависимостями для CoffeeScript, less, CJS/AMD модулей и т.д. для целей разработки;
- перехват фоновых сервисов;
- кастомная шаблонизация, основанная на определенных паттернах URL;
- улучшение производительности, к примеру, предварительная загрузка ресурсов, которые понадобятся пользователю в ближайшем будущем, как несколько последующих картинок в фотоальбоме.

В будущем *serviceworkers* будут способны на многие другие полезные вещи для web-платформ, приблизив их к жизнеспособности нативных приложений. Примечательно, что другие спецификации могут и будут использовать контекст *serviceworker*, например, для следующих целей:

- фоновой синхронизации: запускать *serviceworker*, даже когда ни одного пользователя нет на сайте, чтобы обновить кеш;
- реакции на пуш-сообщения: запускать *serviceworker* для отправки сообщений пользователям, чтобы оповестить их о новом доступном контенте;
- реакции на определенное время и дату;
- введение гео-ограничений.

Также некоторые PWA используют архитектурный подход, который называется AppShellModel. Для мгновенной загрузки сервис воркеры хранят базовый пользовательский интерфейс или «оболочку» (shell) приложения. Данная оболочка является статическим фреймом или макетом, в который контент загружается постепенно, при этом динамически позволяя пользователям взаимодействовать с приложением вне зависимости от качества соединения. Технически оболочка представляет собой код, который хранится в кэше браузера мобильного устройства.

Таким образом, стандартный процесс работы с прогрессивным веб-приложением заключается в следующем: пользователь, используя смартфон, открывает ссылку, полученную посредством любого приложения, после загрузки web страницы пользователь получает полноценное приложение, которое он может использовать.

Тем самым начать использовать новое приложение становится намного проще, так как не нужно заходить в магазин приложений и ждать пока установится необходимое пользователю приложение.

При необходимости пользователь может добавить приложение на рабочий стол посредством пункта опций в браузере «Addtohomescreen». Или приложение может предложить пользователю сделать это вместо него посредством Webapp-install-banner. После чего на главном экране пользователя будет создана иконка, предварительно указанная в манифесте приложения. Разработчик может управлять стилем браузера и вариантом отображения приложения после установки. Также ServiceWorker, идущий в комплекте с PWA приложением, будет отвечать за кеширование, оффлайн работу и обработку пуш сообщений.

Заключение

Прогрессивные веб-приложения являются инновационным подходом к разработке, так как включают в себе сильные стороны веб-сайтов (сами по себе ими и являются) и нативных мобильных приложений. То есть, с одной стороны, приложение имеет ссылочную структуру, небольшой размер, с другой – приложение ведет себя как настоящее нативное приложение с удобным и понятным интерфейсом. В основе PWA лежит ServiceWorker – посредник между устройством пользователя и сетью, который предоставляет контент вне зависимости от качества соединения и его состояния, при этом также предоставляет функции нативных приложений, такие как работа оффлайн, push-уведомления, экраны загрузки и другие.

На момент проведения исследований полная поддержка PWA обеспечивалась во всех браузерах, кроме Safari от Apple. С большей вероятностью данная технология не будет принята компанией Apple. Это может означать, что в ближайшем будущем может появиться новый подход в веб-разработке как конкурент прогрессивных приложений.

ЛИТЕРАТУРА

1. ProgressiveWebApps: WhoAmI[Электронный ресурс]. – Россия, 2017. – Режим доступа: <https://habrahabr.ru/post/303626/>. – Дата доступа: 03.10.2017.
2. Веб-документация MDN [Электронный ресурс] / Манифест веб-приложения. – США, 2017. – Режим доступа: <https://developer.mozilla.org/ru/docs/Web/Манифест>. – Дата доступа: 03.10.2017.
3. Веб-документация MDN [Электронный ресурс] / ServiceWorker API. – США, 2017. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/API/Service_Worker_API. – Дата доступа: 03.10.2017.

УДК 004.42+371.263

АНАЛИЗ СЛОЖНОСТИ ВОПРОСА В СИСТЕМЕ ДЛЯ ПРОВЕДЕНИЯ АДАПТИВНОГО ТЕСТИРОВАНИЯ

С.С. ВОЙТЕХОВИЧ*(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются основные принципы работы системы анализа сложности заданий в рамках системы для проведения адаптивного тестирования. Описано назначение подобной системы, возможный вариант реализации, достоинства и недостатки. Также показаны преимущества, получаемые в результате внедрения алгоритма анализа сложности в систему адаптивного тестирования.

Адаптивное тестирование – это способ контроля знаний, в котором совмещены достоинства оценки знаний с помощью компьютера и преподавательский подход контроля знаний. То есть во время проверки знаний в рамках устного опроса экзаменатор может отталкиваться от уровня знаний экзаменуемого, тем самым подобрав тот уровень сложности вопросов, на который экзаменуемый сможет ответить. Таким образом, под адаптивным тестированием понимается процесс проверки знаний, в рамках которого система способна регулировать сложность заданий в зависимости от уровня подготовленности тестируемого [1].

Для реализации такого процесса могут использоваться разные алгоритмы, но ключевой особенностью каждого из них является оперирование критерием сложности вопроса. Он необходим, так как система чаще всего не способна по формулировке вопроса определить его сложность, и ей требуется некоторое значение, от которого можно отталкиваться в процессе подбора нового набора заданий для тестируемого. Более того, возможность грамотно оперировать данным значением и отличает систему адаптивного тестирования от обычной системы компьютерного тестирования.

Способы определения сложности заданий

Потенциально самым простым способом получения значения сложности для системы является непосредственно составитель тестов. Тесты, так или иначе, должны быть кем-то созданы, и значение сложности для каждого конкретного вопроса может быть указано составителем заданий. Этой информации зачастую будет достаточно для правильной работы алгоритма адаптивного тестирования. Но данный подход имеет два недостатка:

- 1) необходимость составителю тестов анализировать каждый отдельный вопрос и грамотно оценивать его сложность;

- 2) возможная необъективность выбранной сложности для конкретного вопроса.

Говоря о второй проблеме, если в рамках устного опроса экзаменатор выберет неверную сложность вопроса для конкретного экзаменуемого, это создаст сложность только для одного человека. В случае же автоматизированной системы данная проблема может повлиять на прохождение неограниченного количества тестов.

Однако решение данной проблемы можно почерпнуть из того же примера с устным опросом. Экзаменатор, опираясь на ответы экзаменуемых, даже не осознавая сложности вопроса, со временем может ее оценить. То есть решением проблемы с необъективностью оценки сложности вопроса может стать возможность тестирующей системы анализировать результаты ответов на каждое конкретное задание. Потенциально, это же может стать вариантом и для решения первой проблемы. При создании набора тестовых заданий для каждого вопроса может быть указано некоторое значение сложности по умолчанию. В дальнейшем, с каждым новым ответом на конкретный вопрос его сложность будет меняться, и в результате будет получен набор заданий с градацией по сложности.

Таким образом, оценка сложности задания будет основываться не только на опыте и предпочтениях экзаменатора или составителя тестов, но и на реальных данных о прохождении тестов, что потенциально является более точным способом определения сложности заданий.

Вариант реализации системы анализа сложности

Система анализа сложности заданий может быть реализована по-разному. Можно учитывать разные параметры, такие как количество правильных/неправильных ответов, результаты тестируемого по прохождению других тестов по данному предмету (если система хранит такую информацию), время, суммарно затраченное на выполнение задания и другое.

Для примера можно рассмотреть систему, для которой показатель сложности вопроса определяется некоторым значением от 1 до 10, где 1 – самые легкие задания, 10 – самые сложные. При составлении теста некоторому вопросу был задан средний уровень сложности, то есть значение 5. После некоторого количества прохождений данного задания тестируемым имеем соотношение правильных ответов к неправильным, как 4 к 6-ти, то есть на данный вопрос было дано больше неправильных ответов, чем пра-

вильных. На основании этих данных легко определить, что сложность данного задания можно поднять до значения 6.

Потенциально, имея набор тестовых заданий с одинаково равным уровнем сложности, уже после первого его использования мы получаем градацию вопросов с двумя или тремя (в случае, если были показаны не все вопросы) уровнями сложности. И с каждым новым прогоном данного тестового набора количество градаций будет увеличиваться [2].

Рассмотренный вариант потенциально является самым простым способом реализации системы оценки сложности заданий на основании данных по ответам. Он не лишен недостатков как с концептуальной точки зрения, так и с точки зрения программной реализации.

С концептуальной точки зрения, по возможности, должны соблюдаться следующие требования:

- 1) тестируемые не должны знать о системе оценки сложности заданий;
- 2) тестируемые должны быть заинтересованы в получении наивысшей оценки.

Данные требования обусловлены тем, что, зная принцип работы системы, тестируемые могут напавленно пытаться контролировать предоставляемую выборку заданий, намеренно ошибаясь в некоторых вопросах, тем самым повышая их сложность и снижая вероятность их появления в будущем. Такой вариант весьма маловероятен, ибо в таком случае многие в результате тестирования могут получить плохую оценку, но возможен.

С точки зрения реализации данный алгоритм требует хорошо продуманного подхода к организации совместного доступа к данным. То есть при активном использовании системы тестирования, одно и то же задание одновременно может быть показано нескольким тестируемым, и нужно предусмотреть возможность попытки единовременного изменения сложности по результатам нескольких тестов.

Также может понадобиться реализация повторного использования заданий в рамках нескольких тестовых наборов с целью повторного использования информации о его сложности и получения большего количества данных для ее изменения. Это требование имеет и обратную сторону – может понадобиться предоставить возможность использования одного и того же задания с разной и независимой сложностью.

Заключение

Хотя вышеописанные недостатки и имеют место, существует малая вероятность того, что они могут создать большие проблемы для работы адаптивных алгоритмов в целом. Более того, работа адаптивного алгоритма тестирования и алгоритма подбора сложности для задания никак прямо не зависят и попросту используют результаты работы друг друга. Следовательно, система подбора сложности задания может разрабатываться как дополнительная функция системы адаптивного тестирования с возможностью отключения этой функции в случае необходимости. Аналогично, система анализа сложности может использоваться и без адаптивных алгоритмов, чтобы анализировать сложность вопроса для дальнейшего определения их «стоимости» при выставлении отметки или для предоставления дополнительной информации экзаменатору.

При своей достаточно несложной реализации система анализа сложности может упростить работу с системой адаптивного тестирования в целом. Более того, при необходимости, с помощью данной системы, можно полностью скрыть работу адаптивных алгоритмов и от составителя тестов, освобождая последнего от необходимости дополнительного изучения данного вопроса и указания сложности заданий. В итоге можно получить систему с понятным механизмом добавления заданий в базу данных и преимуществами адаптивных алгоритмов.

ЛИТЕРАТУРА

1. Словарь терминов [Электронный ресурс]. – Режим доступа: <http://www.profcareer.ru/lib/dicty.php>. – Дата доступа: 20.09.2017.
2. Трудность теста и тестовых заданий [Электронный ресурс]. – Режим доступа: <http://testolog.narod.ru/Theory7.html>. – Дата доступа: 21.09.2017.

УДК 378.146.004.4

ИСПОЛЬЗОВАНИЕ ЭЛЕКТРОННЫХ СИСТЕМ ОБУЧЕНИЯ ДЛЯ КОНТРОЛЯ ЗНАНИЙ

С.С. ВОЙТЕХОВИЧ*(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются современные электронные системы контроля знаний, их характеристики и возможности использования в системе образования. Также подробно описан процесс применения систем на практике, их основные плюсы и минусы.

В век широкого распространения различных информационных и технических ресурсов все еще используются принципы контроля знаний, разработанные много десятилетий назад. Тем не менее в последнее время начали разрабатывать и новые системы, более подходящие для современной системы образования, которая приобретает все более массовый характер. Одним из самых распространенных нововведений является тестирование, которое позволило произвести переход с бумажных источников на электронные системы обучения.

Как правило, электронные системы обучения внедряются в учреждениях образования для организации дистанционного (или заочного) обучения, но в последнее время все чаще данную систему используют и для аудиторной проверки знаний. Данные системы имеют ряд очевидных плюсов: мобильность (переносимость), вариативность, простота проверки решений, беспристрастное выставление отметок.

Мобильность (переносимость)

В связи с тем, что большинство ресурсов, представляющих собой системы контроля знаний, позволяют проводить не только тестирование в компьютерных классах, во время занятий, но и внеаудиторное регулярное тестирование. Например, студенту или ученику создается аккаунт на ресурсах, подобных x-TLS [1], INDIGO [2], Let'stest [3] или УСАТИК (Универсальная Система Автоматизированного Тестирования и Контроля) [4; 5], и сообщаются учетные данные, необходимые для авторизации. Далее преподаватель выкладывает учебный материал, по которому будет проводиться тестирование, создает тест по этому материалу, а затем выставляет этот тест для выполнения и указывает дату и время, до которых данное задание необходимо выполнить, а также максимальное количество попыток. Кроме того, в некоторых системах можно с помощью календаря управлять временем не только сдачи теста, но и временем появления этих тестов у учеников. Таким образом, организовывается отложенное тестирование.

Возможность указания времени появления теста и срока его сдачи позволяет ученикам выполнять задания в наиболее удобной для них обстановке, в максимально подходящее время. Так как удаленные системы работают круглосуточно, пройти тест можно как ранним утром, так и поздним вечером. Кроме того, ученики, у которых возникли проблемы со здоровьем или другие личные затруднения, могут пройти тест на домашнем компьютере, или даже со своего мобильного телефона в тот момент, когда это будет им удобно. Помимо прочего, если ученик заранее знает, когда появятся новые тесты и к какому сроку их необходимо сдать, это делает его более организованным, учит правильно распределять свое время на учебу и личные дела, а также позволяет качественно и без спешки подготовиться к выполнению задания.

Возможность указания количества попыток прохождения теста есть не в каждой системе, но она полезна. Таким образом, учащийся для того, чтобы пересдать плохой тест, может потратить некоторое время на повторное изучение материала и сразу же приступить к очередной попытке прохождения. Во-первых, это позволяет отстающим ученикам набрать хотя бы минимальный положительный балл. Во-вторых, стимулирует учащихся к повторному прочтению нового и повторению ранее изученного материала. В-третьих, преподавателю в таком случае нет необходимости составлять дополнительные задания для отстающих, так как даже после неудачного прохождения тестов ученик должен иметь возможность реабилитироваться.

Кроме вышеперечисленных ресурсов есть также программы, рассчитанные на проведение тестирования в компьютерных классах. У таких программ либо есть сервер с заданиями и клиенты, в которых авторизуются ученики (связь между ними осуществляется посредством локальной сети), либо для каждого теста генерируется отдельный модуль, который копируется на компьютеры учеников (связи между модулями нет). К таким программам можно отнести Айрен [6], OpenTEST 2.0 [7] и MyTestXPro [8]. Все эти программы могут быть использованы только для аудиторного тестирования, но это не является большим минусом, так как эти системы имеют все достоинства автоматизированных систем тестирования, а выполнение контрольных и экзаменационных тестов в любом случае должно проходить в аудитории под контролем преподавателя.

Вариативность

Как правило, тесты содержат в себе набор типовых вопросов и составление нескольких вариантов заданий представляет собой очень трудоемкую задачу. С другой стороны, если всем ученикам дать одинаковый набор заданий, то повышается возможность списывания, использования шпаргалок, а также с каждой новой попыткой понижается эффективность прохождения тестов. Ведь если ученик знает, на какие вопросы он ответил верно, он может просто повторить свои ответы, не вдумываясь в содержание вопроса и не анализируя его. В то же время, если ученик знает, какие именно вопросы будут в тесте, он акцентирует свое внимание на учебном материале, связанном именно с этими вопросами, и игнорирует другие, порой не менее важные вопросы. Из-за этого зачастую и появляются небольшие пробелы в знаниях, которые могут потом иметь катастрофические последствия.

Решением данной проблемы является такое свойство автоматизированных систем проверки знаний, как вариативность. Преподаватель создает набор вопросов, охватывающий всю предметную область указанной темы, а система, в свою очередь, выбирает из данного набора некоторое конкретное количество заданий и предлагает их ученику в произвольном порядке для выполнения. Такой подход заставляет учеников учить весь теоретический материал, провоцирует к пониманию темы, а не заучиванию номеров правильных ответов, а также гарантирует адекватность полученной оценки реальным знаниям ученика.

Простота проверки решений

Как правило, довольно большой проблемой для преподавателя является проверка результатов тестирования. В случае если тест проходит небольшое количество человек, проверить все тесты вручную не составляет особого труда, но если группа состоит из 20–30 человек, а тест состоит хотя бы из 10–15 вопросов из разных вариантов заданий, проверка всех решений превращается в длительный и трудоемкий процесс, в котором велик процент ошибок со стороны преподавателя, так как со временем человек устает, и его внимание рассеивается. В случае использования автоматизированных систем контроля знаний правильность ответа на вопрос проверяется автоматически, а после прохождения теста результаты видит не только преподаватель, но и ученик. Благодаря системе студент может самостоятельно отслеживать тенденции изменения качества знаний по предмету и оперативно получать информацию о только что пройденном тесте. Системы работают автоматически, и сбоя в выявлении ошибок быть не может, в отличие от проверки вручную.

Беспристрастное выставление отметок

Довольно большой проблемой для преподавателей зачастую становится процесс выставления отметок. В случае устного опроса или письменного экзамена сложно оценить знания ученика, так как существует масса критериев выставления отметок, и всех их нужно придерживаться. Тесты являются отличным решением данной проблемы. Если у преподавателя существует достаточный набор тестовых заданий по данной теме и хорошо продумана система оценивания каждого вопроса, то остается лишь корректно распределить, за какой диапазон баллов результата какая отметка ставится, и система расставит их автоматически. Некоторые системы позволяют выставять различную «стоимость» вопросов, например: легкий вопрос – 1 балл; средний – 2 балла; сложный – 3 балла.

Система также решает проблему возможности влияния личных отношений между учениками и преподавателем на результат проверки знаний. Система не имеет никаких личностных качеств и не учитывает сторонние факторы во время выставления отметок.

Заключение

На основании вышеизложенного материала можно констатировать, что автоматизированные системы оценки знаний – удобное и эффективное средство для проведения как промежуточного, так и итогового контроля знаний. Их использование значительно упрощает и ускоряет работу преподавателя, в то же время стимулирует учеников к наиболее подробному и всестороннему изучению предложенного преподавателем материала, подталкивает их к дальнейшему развитию самодисциплины и самоорганизации.

Единственный минус данных систем – отсутствие возможности проверки заданий творческого характера, таких как рисунки, сочинения, пересказы и тому подобное, но для этого разработаны особые методики проверки, которые нельзя привести к какому-либо четкому своду правил. Если бы творческие задания оценивались системой (даже использующей нейронные сети), результат был бы непредсказуемым.

ЛИТЕРАТУРА

1. Автоматизированная система обучения и контроля знаний на основе расширенных тестовых заданий x-TLS [Электронный ресурс] / О программе для тестирования знаний и конструкторе тестов x-TLS. – Киев, 2016. – Режим доступа: <http://xtls.org.ua/about.php.html>. – Дата доступа: 27.12.2016.

2. INDIGO [Электронный ресурс] / Программа для создания тестов и онлайн тестирования. – Ставрополь, 2016. – Режим доступа: <http://indigotech.ru>. – Дата доступа: 27.12.2016.
3. Система тестирования Let'stest [Электронный ресурс] / Конструктор для создания тестов. Онлайн сервис для проверки знаний учащихся и аттестации работников. – Воронеж, 2016. – Режим доступа: <https://letstest.ru>. – Дата доступа: 27.12.2016.
4. Компьютерная газета A-Z [Электронный ресурс] / УСАТИК – система тестирования знаний. – Минск, 2016. – Режим доступа: <http://www.nestor.minsk.by/kg/2006/29/kg62917.html>. – Дата доступа: 28.12.2016.
5. УСАТИК [Электронный ресурс] / Описание пакета УСАТИК. – Ульяновск, 2001. – Режим доступа: <http://www.usatic.narod.ru>. – Дата доступа: 28.12.2016.
6. Айрен [Электронный ресурс] / Программа тестирования знаний. – М., 2009. – Режим доступа: <http://irenproject.ru>. – Дата доступа: 05.01.2017.
7. Программа тестирования [Электронный ресурс] / Компьютерное тестирование студентов OpenTEST 2. – Киев, 2013. – Режим доступа: <http://opentest.com.ua/kompyuternaya-programma-testirovaniya-znaniy-opentest-2>. – Дата доступа: 05.01.2017.
8. Компьютерное тестирование знаний и компьютерные педагогические тесты [Электронный ресурс] / MyTestXPro – система программ для создания и проведения компьютерного тестирования, сбора и анализа их результатов. – Москва, 2016. – Режим доступа: <http://mytest.klyaksa.net>. – Дата доступа: 05.01.2017.

УДК 004.021

ФРЕЙМВОРК SEMANTICUI

А.И. ПИРОВИЧ

(Представлено: Д.В. ПЯТКИН)

Рассматривается современный фреймворк SemanticUI для разработки веб-приложений. Показана его уникальность и особенности отличия от других фронт-енд фреймворков.

SemanticUI – современный фреймворк для фронт-енд разработки веб-приложений, основанный на LESS и jQuery; имеет изящный, утонченный и адаптивный дизайн, который обеспечивает гибкий пользовательский интерфейс.

Semantic UI предлагает несколько способов изменения элементов пользовательского интерфейса. Для крупных проектов, которые основаны на создании персонализированного визуального языка, ориентированного на бренд, темы сайта позволяют изменять базовые переменные, управляющие SemanticUI, а также определять альтернативный переопределяющий CSS. Темы сайтов переносимы между проектами и влияют на скомпилированный код фреймворка.

Для небольших проектов, проектов с быстрыми сроками упакованные темы отлично подходят для заимствований из других проектов с открытым исходным кодом в этой области.

Для любого типа проекта важным местом для начала настройки Semantic UI является файл `site.variables` – файл переменных, который наследует все другие файлы переменных.

Семантический интерфейс уникален по двум причинам: *во-первых*, фреймворк структурирован; он использует пять описательных категорий для определения повторно используемых компонентов пользовательского интерфейса. Организованные вместе на странице создают визуальное представление, которое мы считаем веб-сайтом.

- **UIElement** является базовым строительным блоком. Он может использоваться в одиночку либо в группах. Например, кнопка может быть независимой или помещаться в группу кнопок;

- **UICollection** – это группа различных элементов, которые являются зависимыми друг от друга. Например, веб форма может иметь кнопки, инпуты, иконки и т.д.;

- **UI View** представляет общую часть контента веб-сайта. Это может быть раздел новостей, секция комментариев;

- **UI Module** – модули, с интерактивной функциональностью на основе JavaScript. Примерами служат аккордеон, попап окна, поиск и т.д.;

- **UI Behavior** – компонент, который не может существовать независимо. Он используется для добавления функциональности в другие компоненты. Например, Form Validation предусматривает проверку для компонента Form.

Почти у каждого компонента есть типы, состояния и варианты использования. Например, для компонента `button` некоторые из типов включают в себя: стандартную кнопку, кнопку со значком, анимированную кнопку. Также кнопка может находиться в активном, отключенном или загруженном состоянии. Наконец, кнопка может отличаться по размеру и цвету и может быть отформатирована как базовая, социальная, переключающая и др. Эта система дает вам большую гибкость при внешнем виде компонента.

Semantic UI не только значим и хорошо структурирован с точки зрения именования его классов, но также именования, определения и описания его компонентов. Его структура гораздо более смысловая, чем в Bootstrap или Foundation.

Во-вторых, уникальность семантического интерфейса заключается в том, что она предоставляет некоторые особые функции и компоненты, не присутствующие в других фреймворках. Например, `Feed` и `Comment` в `UI Views components` или `Sidebar` и `Shape` в `UI Modules`. Кроме этого, при взаимодействии с компонентами SemanticUI имеется возможность отладки в режиме реального времени. Просто откройте свою веб-консоль, и вы будете видеть, что точно передают ваши компоненты, что они делают.

Еще одна сильная сторона Semantic UI заключается в том, что он использует минимальный и нейтральный стиль, оставляя настройки открытыми для вас. Он включает важные и полезные вещи, игнорируя дополнительные функции, которые вы, вероятно, никогда не будете использовать. Кроме того, компоненты фреймворка являются переносимыми и автономными, поэтому вы можете взять и использовать только те, которые вам нужны.

В фреймворке используются элементы `em` и `rem` для своих элементов, что делает его полностью отзывчивым и адаптивным для любого размера. Вам нужно только изменить базовый шрифт, и все остальные элементы будут соответствующим образом отрегулированы.

Заключение

Подводя итог, отметим, что Semantic UI хорошо документирован, веб-сайт содержит множество примеров для разных компонентов.

Как можно видеть, SemanticUI является новым и в некоторых аспектах уникальным фреймворком, который хорошо документирован на английском языке. Официальный веб-сайт содержит множество примеров для разных компонентов. Кроме того, в нем есть руководство по стилю с методами и указаниями о том, как писать код.

ЛИТЕРАТУРА

1. SemanticUI [Электронный ресурс]. – Режим доступа: <https://semantic-ui.com/>. – Дата доступа: 26.09.2017.
2. SemanticUIIntroduction [Электронный ресурс]. – Режим доступа: <http://learnsemantic.com/developing/customizing.html#the-future-of-themes>. – Дата доступа: 26.09.2017.
3. Semantic UItutorials [Электронный ресурс]. – Режим доступа: <https://github.com/Semantic-Org/Semantic-UI/wiki/Tutorials>. – Дата доступа: 26.09.2017.

УДК004.021

ТЕХНОЛОГИЯ FLEXBOX

А.И. ПИРОВИЧ

(Представлено: Д.В. ПЯТКИН)

Рассматривается новая CSS технология Flexbox, ее основные преимущества и актуальность. Это технология позволяет заменить float и табличную разметку. Она поставляется с умным набором значений по умолчанию, которые упрощают создавать сложное форматирование с относительно небольшим кодом.

Долгое время единственными надежными кросс-браузерными инструментами, доступными для создания макетов CSS, были такие вещи, как float и position. Это прекрасно, и они работают, но в некотором смысле они также довольно ограничены.

Следующие требования трудно, иногда невозможно, достичь с помощью этих инструментов удобным и гибким способом:

- вертикальное центрирование содержимое блока внутри его родителя;
- создание всех дочерних элементов контейнера занимает равное количество доступной ширины / высоты независимо от ширины и высоты;
- создание макета со столбцами, в котором они имели одинаковую высоту, даже если содержат различный объем контента.

Использование Flexbox значительно облегчает эту работу.

Создание всех детей контейнера занимает равное количество доступной ширины / высоты независимо от ширины и высоты.

Создание всех столбцов в макете с несколькими столбцами допускает одинаковую высоту, даже если они содержат различное количество контента.

Существует три версии спецификации, но только одна из них действительно имеет значение «display: flex» с отличной поддержкой браузера.

Основными преимуществами flexbox являются:

- все блоки очень легко делаются «резиновым», что уже следует из названия «flex». Элементы могут сжиматься и растягиваться по заданным правилам, занимая нужное пространство;
- выравнивание по вертикали и горизонтали, базовой линии текста работает шикарно;
- расположение элементов в html не имеет решающего значения. Его можно поменять в CSS;
- это особенно важно для некоторых аспектов responsive верстки;
- элементы могут автоматически выстраиваться в несколько строк/столбцов, занимая все предоставленное место;

- множество языков в мире используют написание справа налево rtl (right-to-left), в отличие от привычного нам ltr (left-to-right). Flexbox адаптирован для этого. В нем есть понятие начала и конца, а не права и лева. То есть в браузерах с локалью rtl все элементы будут автоматически расположены в реверсном порядке.

Макет Flexbox состоит из flex container, который содержит flex items. Flex container имеет главную и поперечную ось. Направление главной оси задает свойство flex-direction. А поперечная ось всегда будет перпендикулярна главной (рис. 1).

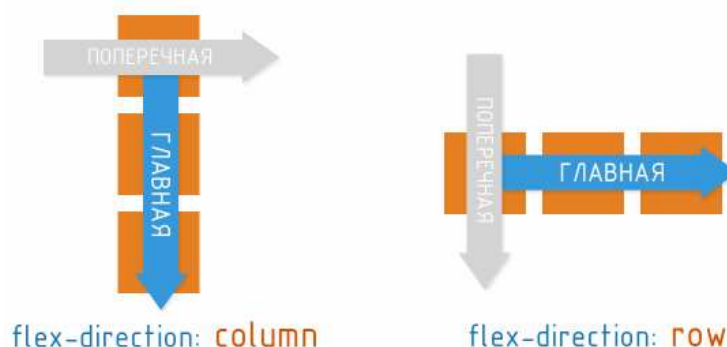


Рисунок 1. – Главная и поперечная ось Flexконтейнера

Flexbox включает в себя набор интеллектуальных значений по умолчанию, поэтому вы можете создать мощный макет с очень маленьким кодом. Пример представлен на рисунке 2.

```
.flex-container {
  display: flex;
}
```

Рисунок 2. – Пример использования flexbox

Эта одна строка кода выполняет все следующие действия:

- делает .flex-container как гибкий контейнер;
- рассматривает все дочерние элементы .flex-container как гибкие предметы;
- элементы Flex будут выложены в горизонтальной линии;
- элементы Flex будут размещены в исходном порядке;
- элементы Flex будут выложены с левой стороны гибкого контейнера;
- элементы Flex будут определяться на основе их обычных свойств width;
- если для всех элементов гибкости недостаточно места, им будет разрешено сжиматься горизонтально до тех пор, пока они не подойдут;
- если им нужно сжиматься, каждый элемент будет уменьшаться одинаково;
- элементы Flex будут растягиваться вертикально, чтобы соответствовать высоте самого высокого элемента.

Отсюда видно, насколько мощным может быть Flexbox без дополнительной работы.

Текущая спецификация Flexbox полностью поддерживается во всех современных браузерах, включая мобильные, IE11 и Edge (рис. 3). IE10 также поддерживает Flexbox, но использует tweenserspec (display:flexbox).

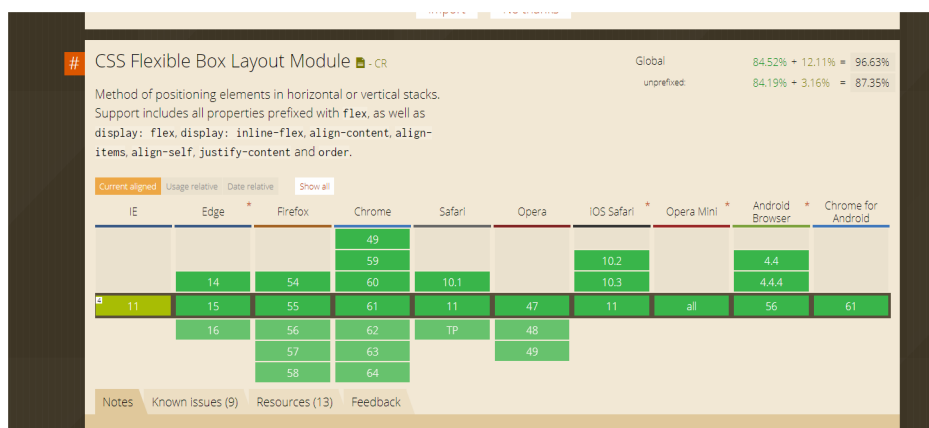


Рисунок 3. – Поддержка браузерами

Заключение

Flexbox – это новая технология, которая позволяет заменить float и табличную разметку. Она поставляется с умным набором значений по умолчанию, которые упрощают создавать сложное форматирование с относительно небольшим кодом. Flexbox стабильна, проста в использовании и хорошо поддерживаемая технология.

ЛИТЕРАТУРА

1. Whatisflexboxtechnology [Электронный ресурс]. – Режим доступа: <http://spaceninja.com/2015/08/24/what-is-flexbox/>. – Дата доступа: 26.09.2017.
2. Learnflexbox [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox. – Дата доступа: 26.09.2017.
3. Guidetoflexbox [Электронный ресурс]. – Режим доступа: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>. – Дата обращения: 26.09.2017.
4. Flexbox [Электронный ресурс]. – Режим доступа: <http://html5.by/blog/flexbox/>. – Дата доступа: 26.09.2017.
5. CanIUseFlexbox [Электронный ресурс]. – Режим доступа: <http://caniuse.com/>. – Дата доступа: 26.09.2017.

УДК 004.6

**МОДЕЛИРОВАНИЕ ДАННЫХ В NOSQL СУБД
НА ПРИМЕРЕ РАЗРАБОТАННОГО ВЕБ-ПРИЛОЖЕНИЯ
«ИНФОРМАЦИОННЫЙ АГРЕГАТОР БАНКОВ РЕСПУБЛИКИ БЕЛАРУСЬ»**

Р.П. ГУСЕВСКИЙ*(Представлено: Д.В. ПЯТКИН)*

Проанализированы отличия моделирования в NoSQL и SQLСУБД на примере разработанного веб-приложения «Информационный агрегатор банков Республики Беларусь», созданного с использованием одной из популярных СУБД MongoDB.

К настоящему времени человечеством накоплено поистине гигантское количество информации об объектах и явлениях. Но эта информация не лежит мертвым грузом, она хранится в электронном виде и используется в базах данных. Базы данных – это часть информационных систем – программно-аппаратных комплексов, осуществляющих хранение и обработку огромных информационных массивов.

База данных представляет собой определенным образом структурированную совокупность данных, совместно хранящихся и обрабатывающихся в соответствии с некоторыми правилами. Как правило, база данных моделирует некоторую предметную область или ее фрагмент. Программа, производящая манипуляции с информацией в базе данных, называется СУБД (система управления базами данных). Основными составляющими информационных систем, построенных на основе баз данных, являются файлы БД, СУБД и программное обеспечение (клиентские приложения), позволяющие пользователю манипулировать информацией и совершать необходимые для решения его задач действия. Базы данных как способ хранения больших объемов информации и эффективного манипулирования ею используются практически во всех областях человеческой деятельности.

Выбор NoSQLСУБДMongoDB

Для того чтобы хранить нужные данные, следует определиться с выбором БД, для реализации информационного агрегатора была выбрана база данных MongoDB, которая хорошо интегрируется с веб-приложениями. Это обусловлено тем, что MongoDB хранит данные в формате JSON. Формат JSON имеет хорошую поддержку в Javascript, поэтому работа с JSON хорошо отразится на производительности данного приложения.

MongoDB является NoSQL базой данных. NoSQL базы данных не работают с реляционными моделями. Существует много различных решений, каждое из которых работает по-своему и служит специфической цели. Эти безсхемные решения снимают ограничения с формирования сущностей и допускают хранения данных в виде ключ-значение.

NoSQL базы данных не используют общий формат запроса, такой как SQL в реляционных базах данных. Каждое NoSQL решение использует собственную систему запросов.

В MongoDB данные имеют гибкую схему хранения документов в одной коллекции. Это означает, что документы не должны иметь одинаковый набор полей или структуру. Общие поля в документах коллекции могут хранить различные типы данных [1].

Для того чтобы моделировать данные в MongoDB, нужно:

1. Комбинировать объекты в одном документе, если есть необходимость использовать их вместе.
2. Дублировать данные, так как время вычислений «дороже», чем место на диске.
3. Объединять данные во время записи, а не во время чтения.
4. Использовать сложные агрегации в схеме.
5. Оптимизировать схему под наиболее частые случаи.

Наглядная разница моделирования данных в NoSQL от реляционной базы данных

Чтобы более детально понять, в чем разница между моделированием в NoSQL базы данных и реляционной БД, рассмотрим следующий пример [2].

Предположим, нужно создать базу данных для хранения проектов и разработчиков, которые работают над этими проектами. Следует учесть, что каждый проект имеет уникальное имя и описание, и имеет много разработчиков.

Если использовать стандартную реляционную БД, то наша структура таблиц, примерно, имела вид, который представлен на рисунке 1.

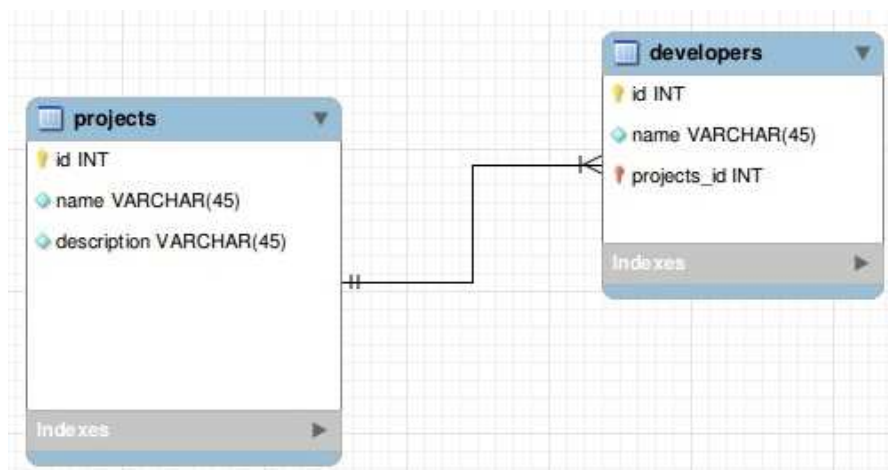


Рисунок 1. – Структура таблиц, рассматриваемая в примере

В MongoDB данная схема выглядела бы в виде структуры в формате JSON (листинг 1).

Листинг 1 – Пример схемы в MongoDB

```

{
  _id: PROJECT_ID
  name: NAME_OF_PROJECT,
  developers: [
    {
      name: 'DEVELOPER_NAME',
    },
    {
      name: 'DEVELOPER_NAME'
    }
  ]
}
  
```

Анализ и моделирование данных при разработке веб-приложения в MongoDB

При анализе функциональности веб-приложения выявлены следующие сущности:

1. Сущность *Обменный пункт* содержит данные о конкретном обменном пункте. Характеризуется наименованием, описанием, координатами: широты, долготы, городом, адресом.
2. Сущность *Отделение* описывает информацию о конкретном отделении банка. Характеризуется наименованием, описанием, координатами: широты, долготы, городом, адресом.
3. Сущность *Банкомат* содержит конкретную информацию о банкомате, принадлежащем конкретному банку. Характеризуется адресом, точной локацией, наименованием, временем работы, возможной валютой для снятия денег, координатами: широты, долготы.
4. Сущность *График Валют* содержит информацию для построения графиков валют. Характеризуется объектами валюты.
5. Сущность *Валюта* содержит описание валюты. Характеризуется наименованием валюты, ее кодом, массивом продажи и покупки данной валюты.
6. Сущность *Ссылки* содержит полезные ссылки банка. Характеризуется перечнем ссылок для банка.

Опираясь на рекомендации по моделированию данных в MongoDB, а также на результат краткого анализа функциональной части проекта смоделирована общая структура для хранения информации о банке (листинг 2).

Листинг 2 – Общая структура для хранения информации о банке

```
{
  nameBank: "ЗАО «БСББанк»",
  description: "Краткое описание банка",
  shortName: "BSB Bank",
  linkNews: "",
  linkBankCard: "",
  linkCredit: "",
  linkInternetBanking: "",
  linkInsurance: "",
  pathImg: "../img/banksLogo/bsbbank-logo.png",
  affiliate: [],
  exchangeOffice: [],
  bankomats: [],
  graphicCurrency: [],
  mainAddress: "",
  mainNumbers: "+375-17-306-20-40\n+375-29-306-20-40\n"+375-33-306-20-40",
  mainlinkAddress: "https://www.bsb.by/"
}
```

Из описания структуры видно, что каждое поле принимает значение различного типа, так как MongoDB хранит информацию в формате JSON [3], то в данном формате в качестве значений могут быть использованы:

1. Объект – это неупорядоченное множество пар ключ – значение, заключенное в фигурные скобки «{ }». Ключ описывается строкой, между ними значением стоит символ «:». Пары ключ – значение отделяются друг от друга запятыми.

2. Массив (одномерный) – это упорядоченное множество значений. Массив заключается в квадратные скобки «[]». Значения разделяются запятыми.

3. Число.

4. Литералы – true, false и null.

5. Строка – это упорядоченное множество из нуля и более символов, заключенное в двойные кавычки.

Исходя из того, что некоторые поля структуры (объекта) принимали в качестве значений массив, то данные таких полей представляли собой схожие объекты, которые нацелены на выполнение конкретного функционала.

Заключение

Авторами придумана при помощи длительного анализа общая структура, которая представляет собой объект, который описан выше (см. листинг 2). Данная структура хорошо подойдет для использования ее, как и для работы с графиками, так и для использования отображения местоположений банков, отделений и предоставления общей информации о банке. Выбор MongoDB в качестве СУБД приложения выбрана исходя из следующих плюсов:

- данная система хранения данных основана на принципе хранения документов в BSON (Binary JSON) формате, что делает ее хорошо интегрируемой с большей частью веб-приложений;
- более быстрое извлечение простых структур данных;
- хранение неструктурной информации;
- индекс по любому признаку;
- богатые запросы, репликация и высокая доступность [4].

ЛИТЕРАТУРА

1. METANIT.COM Сайт о программировании [Электронный ресурс]. – Режим доступа: <https://metanit.com/nosql/mongodb/1.1.php>. – Дата доступа: 17.09.2017.
2. Руководство по MongoDB. Моделирование данных [Электронный ресурс]. – Режим доступа: http://proselyte.net/tutorials/mongodb/data_modeling/. – Дата доступа: 19.09.2017.
3. Википедия : свободная энцикл. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/JSON/>. – Дата доступа: 19.09.2017.
4. W3IILatestwebdevelopmenttutorials [Электронный ресурс]. – Режим доступа: http://www.w3ii.com/ru/mongodb/mongodb_advantages.html. – Дата доступа: 22.09.2017.

УДК 004.021

**ВЫБОР ТЕХНОЛОГИЙ И РАЗРАБОТКА АДАПТИВНОГО ВЕБ-ПРИЛОЖЕНИЯ
«ИНФОРМАЦИОННЫЙ АГРЕГАТОР БАНКОВ РЕСПУБЛИКИ БЕЛАРУСЬ»****Р.П. ГУСЕВСКИЙ****(Представлено: Д.В. ПЯТКИН)**

Представлен практический способ создания интерфейса для веб-приложения «Информационный агрегатор банков Республики Беларусь». Целью работы явилось написание веб-приложения с удобным интерфейсом, который агрегирует данные с различных официальных банковских сайтов и предоставляет общие сведения по курсам валют в виде графиков, а также новостей и предоставляет ссылки на нужную информацию о банках, информация о местоположениях банкоматов, отделений и обменных пунктов.

Современному человеку представить жизнь без банка очень сложно. Сегодня через банки проходит практически вся наша финансовая жизнь: мы получаем зарплату на банковские карты, берем кредиты, ипотеку, платим через банки за образование детей, оплачиваем услуги ЖКХ, получаем и отправляем денежные переводы, обмениваем денежные средства. Из-за того, что данная тематика всегда будет актуальна, выбрана разработка веб-приложения «Информационный агрегатор банков Республики Беларусь». Немалую роль играют веб-приложения агрегаторы, где консолидируются сведения одновременно из нескольких источников. Они представлены в формате единого пользовательского интерфейса. Такого плана ресурсы крайне полезны при поиске наиболее выгодных условий предоставления услуг и быстрого получения всех необходимых данных. Приложения-агрегаторы – это правильный выбор, когда необходимо быстро найти искомое или сориентировать пользователя на целевые действия [1].

Разработка и выбор технологий для данного веб-приложения

Для реализации приложения были выбраны языки программирования: Javascript, Typescript, программная платформа NodeJS, фреймворк Angular 2, языком разметки веб-страниц является HTML, а также применялись каскадные таблицы стилей (CSS).

Node.js представляет среду выполнения кода на JavaScript, которая построена на основе движка JavaScriptChrome V8, который позволяет транслировать вызовы на языке JavaScript в машинный код. Node.js прежде всего предназначен для создания серверных приложений на языке JavaScript [2].

Для подключения MongoDB к NodeJS использовался фреймворк Express и система управления БД MongoDB для Windows Server 2008 R2 64-bit and later, with SSL support x64.

Express – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. Express позволяет иметь в своем распоряжении множество служебных методов HTTP и промежуточных обработчиков, что позволяет создать надежный API как можно быстрее и легче. Рассматриваемый фреймворк обладает хорошей производительностью за счет тонкого слоя фундаментальных функций веб-приложений, которые не мешают вам работать с функциями Node.js [3].

Чтобы установить связь между API и базой данных, хранящейся в MongoDB, нужно воспользоваться MongoDB Database Server, который устанавливается вместе с системой управления. Для хранения строки соединения с базой данных был создан отдельный модуль db.js (листинг 1).

Листинг 1 – модуль db.js

```
var MongoClient = require('mongodb').MongoClient;

var state = {
  db: null
};

exports.connect = function(url, done){
  if(state.db){
    return done();
  }
  MongoClient.connect(url, function (err, db){
    if(err){
      return done(err);
    }
  })
}
```

```
state.db = db;
  done();
})
};

exports.get = function(){
  return state.db;
};
```

Для лучшей работы с БД разделена логика на модель и контроллер. Создано два модуля: один модуль `banks.js`, который лежит в каталоге `controllers`, а другой модуль с аналогичным названием, который располагается в каталоге `models`.

В модели лежит основная логика для работы с базой данных в MongoDB, а в контроллере – логика для обработки запросов на сервер и получения информации пользователю, который использует данное API. Исходя из того, что модель и контроллер связан, будет предоставлен листинг с работой одного метода как в модели (листинг 2), так и в контроллере (листинг 3).

Листинг 2 – метод получения информации банка в модели

```
var db = require('../db');

exports.getBankByShortName = function(shortName,cb){
  db.get().collection('banks').findOne({ shortName: shortName}, function (err,doc) {
    cb(err,doc);
  });
};
```

Как описывалось ранее, логика контроллера напрямую зависит от логики модели, что описывается в примере, указанном ниже (листинг 3). Здесь установлена связь при помощи метода `require`, который принимает путь модуля и возвращает переменную-связь с данным унаследованным модулем. `Require` дает знать, что данный модуль наследует весь функционал из предыдущего модуля, обозначенный ключевым словом `exports`.

Листинг 3 – метод получения информации банка в контроллере

```
var Banks = require('../models/banks');
exports.getNameBanks = function(req,res){
  Banks.all(function (err, docs){
    if(err){
      console.log(err);
      return res.sendStatus(500);
    }
    docs = docs.map(function(bank){
      return {nameBank : bank.nameBank, shortName: bank.shortName ,pathImg: bank.pathImg};
    });
    res.send(docs);
  });
};
```

На клиентской части использовался фреймворк Angular 2. Он предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, шаблоны, маршрутизация и многое другое [4]. Одной из ключевых особенностей Angular является то, что он использует в качестве языка программирования TypeScript, что хорошо помогает в тестировании отдельных модулей и компонент приложения, так как этот язык, в отличие от Javascript, является строго типизированным.

Для того чтобы создать приложение на Angular 2, нужно установить Angular CLI, данный пакет поможет быстро развернуть и настроить проект. После разворачивания проекта необходимо создать модуль приложения, создать компоненты, разметку и стили для компонент и приступить к работе.

Весь дизайн сайта написан вручную, для достижения адаптивной верстки использован bootstrap 3.

При разработке дизайна клиентской части веб-приложения были учтены следующие *правила*:

1. Использование высококачественных изображений.
2. Существование каждого отдельного элемента на веб-странице должно быть чем-то обусловлено.
3. Изучение сайтов-аналогов.
4. Простота конструкции.
5. Подбор шрифта в сочетании с дизайном, шрифт одна из важнейших частей дизайна.
6. Больше графики, меньше текста.
7. Выделение важной информации.
8. Оптимизация сайта для удобства пользователя.
9. Умение продумывать и заполнять «воздушное пространство» на веб-сайте.
10. Краткая подача материала.

Главная страница с панелью навигацией показана на рисунке 1.

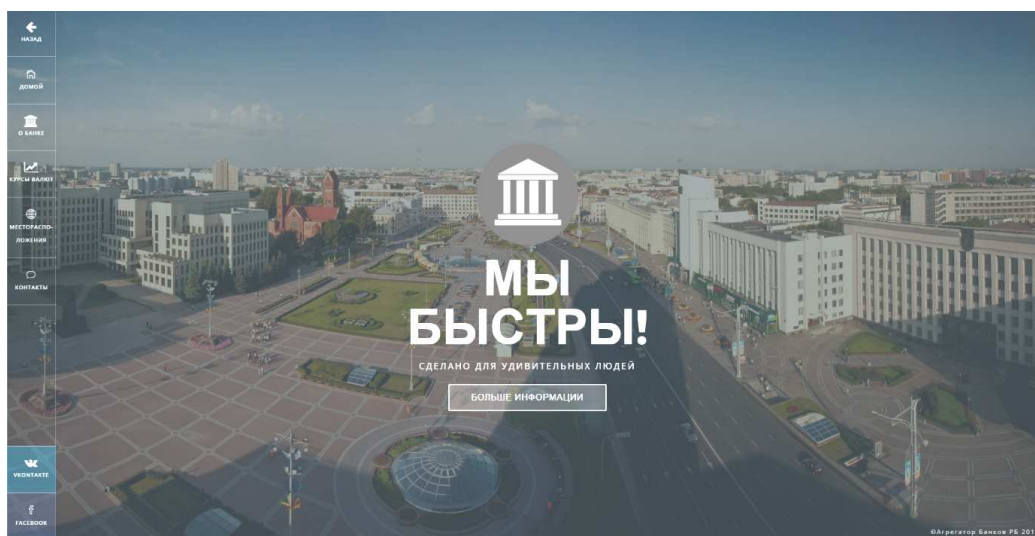


Рисунок 1. – Главная страница с панелью навигацией

Заключение

Авторами разработано веб-приложение с адаптивным дизайном, с серверной частью, которая была хорошо спроектирована и разделена на сущности контроллера и модели, что позволяет пользователю осуществить поиск нужной информации о банке в короткий промежуток времени. Надо всего лишь выбрать нужный банк и на панели навигации выбрать пункт, который нужен пользователю. На данный момент ведутся работы по увеличению функционала программы.

ЛИТЕРАТУРА

1. Мобильная газета бизнеса [Электронный ресурс]. – Режим доступа: <https://mobile.business-gazeta.ru/article/346729>. – Дата доступа: 19.09.2017.
2. METANIT.COM.NodeJS и MongoDB [Электронный ресурс]. – Режим доступа: <https://metanit.com/web/nodejs/6.1.php>. – Дата доступа: 23.09.2015.
3. Express [Электронный ресурс]. – Режим доступа: <http://expressjs.com/ru/>. – Дата доступа: 25.09.2015.
4. METANIT.COM.Руководство по Angular [Электронный ресурс]. – Режим доступа: <https://metanit.com/web/angular2>. – Дата доступа: 25.09.2017.

УДК 004.021

**ОПТИМИЗАЦИЯ АРХИТЕКТУРЫ БАЗ ДАННЫХ
С ЦЕЛЬЮ ПОВЫШЕНИЯ НАДЕЖНОСТИ И СКОРОСТИ РАБОТЫ****К.Х. ГРИГОРЯН****(Представлено: Д.В. ПЯТКИН)**

Рассматриваются технологии и приемы, для реализации отказоустойчивости серверной части приложения и увеличения отклика базы данных. Проанализированы общие приемы оптимизации работы с базами данных.

Довольно часто основной проблемой и упущением при работе с большими объемами данных является плохо организованная часть приложения, отвечающая за хранение данных. Современные технологии требуют повышенного внимания к back-end части приложения. А разработка back-end'a начинается с базы данных. Базы данных, их устройство и архитектура играют ключевую роль в работоспособности приложения и целостности данных пользователей.

Репликация

Репликация позволяет создать полный дубликат базы данных. Так, вместо одного сервера в разрабатываемом продукте будет несколько. Чаще всего используют схему master-slave:

1. **Master** – это основной сервер БД, куда поступают все данные. Все изменения в данных (добавление, обновление, удаление) должны происходить на этом сервере.

2. **Slave** – это вспомогательный сервер БД, который копирует все данные с мастера. С этого сервера следует читать данные. Таких серверов может быть несколько.

Репликация позволяет использовать два или больше одинаковых серверов вместо одного. Операций чтения (SELECT) данных часто намного больше, чем операций изменения данных (INSERT/UPDATE). Поэтому репликация позволяет разгрузить основной сервер за счет переноса операций чтения на Slave.

Для организации технологии репликации, в программном коде продукта необходимо объявить два соединения с базой данных. Одно – для мастера, одно – для Slave. Пример реализации предоставлен в листинге 1.

Листинг 1 – Несколько подключений в виде репликации.

```
<?
$master = mysql_connect('10.10.0.1', 'root');
$slave = mysql_connect('10.10.0.2', 'root');
// исполняемый код
$q = mysql_query('INSERT INTO users ...', $master);
// другая часть приложения
$q = mysql_query('SELECT * FROM users WHERE...', $slave);
```

Репликация обычно поддерживается самой СУБД (например, MariaDb) и настраивается независимо от приложения.

Следует отметить, что репликация сама по себе не очень удобный механизм масштабирования. Причина тому – рассинхронизация данных и задержки в копировании с Master на Slave. Однако это отличное средство для обеспечения отказоустойчивости. Вы всегда можете переключиться на слейв, если мастер ломается и наоборот. Чаще всего репликация используется совместно с шардингом, именно из соображений надежности.

Шардинг

Шардинг – это другая техника масштабирования работы с данными. Суть его в разделении базы данных на отдельные части так, чтобы каждую из них можно было вынести на отдельный сервер.

Есть несколько типов шардинга – вертикальный и горизонтальный.

Горизонтальный шардинг – это разделение одной таблицы на разные сервера. Это необходимо использовать для таблиц, которые хранят внушительные объемы информации и не умещаются на одном сервере. Разделение таблицы на куски делается по такому принципу:

- 1) на нескольких серверах создается одна и та же таблица (только структура, без данных);
- 2) в приложении выбирается условие, по которому будет определяться нужное соединение (например, четные на один сервер, а нечетные – на другой);
- 3) перед каждым обращением к таблице происходит выбор нужного соединения.

Предположим, что есть приложение, которое работает с высоконагруженной таблицей, хранящей фотографии пользователей. Также имеется два сервера (обычно они называются шардами) для этой таблицы. Возможный сценарий использования – для нечетных пользователей хранить и извлекать информацию, используя первый сервер, а для четных – второй. Таким образом, на каждом из серверов будет только часть всех данных о фотографиях пользователей. Пример представлен в листинге 2.

Листинг 2 – Несколько подключений, в стиле горизонтальный шардинг.

```
<?
$image_connections = [
    '1' => '10.10.0.1',
    '2' => '10.10.0.2',
];

$user_id = $_SESSION['user_id'];
// получение фотографий для пользователя $user_id
$connection_num = $user_id % 2 == 0 ? 1 : 2;
$connection =
mysql_connect($image_connections[$connection_num], 'root', '');
$q = mysql_query('SELECT * FROM images WHERE user_id = ' . intval($user_id), $connection);
```

Результат операции $\$user\_id \% 2$ будет остатком от деления на 2. То есть для четных чисел – 0, а для нечетных – 1.

Горизонтальный шардинг – это очень мощный инструмент масштабирования данных.

Вертикальный шардинг – это выделение таблицы или группы таблиц на отдельный сервер.

С целью применения такой разновидности шардинга, как вертикальный, необходимо использовать соответствующее соединение для работы с каждой таблицей.

Пример представлен в листинге 3.

Листинг 3 – Использование нескольких подключений в стиле вертикальный шардинг.

```
<?
$users_connection = mysql_connect('10.10.0.1', 'root', '');
$image_connection = mysql_connect('10.10.0.2', 'root', '');

$q = mysql_query('SELECT * FROM users WHERE ...', $users_connection);
$q = mysql_query('SELECT * FROM images WHERE...', $image_connection);

$q = mysql_query('SELECT * FROM albums WHERE...', $image_connection);
```

В отличие от репликации, мы используем разные соединения для любых операций, но с определенными таблицами.

Не следует применять технику шардинга ко всем таблицам. Правильный подход – это поэтапный процесс разделения растущих таблиц. Следует задумываться о горизонтальном шардинге, когда количество записей в одной таблице является заметным фактором на производительность работы приложения. Шардинг и репликация зачастую используются совместно. Пример такой реализации представлен в листинге 4.

Листинг 4 – Совместное использование шардинга и репликации

```
<?
$image_connections = [
    '1' => [
        'master' => '10.10.0.10',
        'slave' => '10.10.0.11',
    ],
    '2' => [
        'master' => '10.10.0.20',
        'slave' => '10.10.0.21',
    ],
];

$user_id = $_SESSION['user_id'];
```

```
// Читаем данные со слейвов
$connection_num = $user_id % 2 == 0 ? 1 : 2;
$connection = mysql_connect($image_connections[$connection_num]['slave'], 'root', '');
$q = mysql_query('SELECT * FROM image WHERE user_id = ' . intval($user_id), $connection);

//исполняемый код...
//Изменение данных происходит на мастерах
$image_id = 7;
$connection_num = $user_id % 2 == 0 ? 1 : 2;
$connection = mysql_connect($image_connections[$connection_num]['master'], 'root', '');
$q = mysql_query('UPDATE images SET views = views + 1 WHERE image_id = ' . intval($image_id),
$connection);
'2' => '10.10.0.2',
```

Такая схема часто используется не только для масштабирования, но и для обеспечения отказоустойчивости. Так, при выходе из строя одного из серверов шарда, всегда будет запасной.

Заключение

Итогом данной работы можно выделить рассмотренные некоторые возможности для грамотной реорганизации баз данных с целью улучшения оптимизации и увеличения отказоустойчивости.

ЛИТЕРАТУРА

1. Semantic UI [Электронный ресурс]. – Режим доступа: <https://semantic-ui.com/>. – Дата доступа: 26.09.2017.
2. Semantic UI Introduction [Электронный ресурс]. – Режим доступа: <http://learnsemantic.com/developing/customizing.html#the-future-of-themes>. – Дата доступа: 26.09.2017.
3. Semantic UI tutorials [Электронный ресурс]. – Режим доступа: <https://github.com/Semantic-Org/Semantic-UI/wiki/Tutorials>. – Дата доступа: 26.09.2017.

УДК 004.021

АВТОМАТИЧЕСКОЕ ПОЛУЧЕНИЕ СЕРТИФИКАТОВ ДЛЯ ОРГАНИЗАЦИИ ПРОТОКОЛОВ HTTPS

К.Х. ГРИГОРЯН*(Представлено: Д.В. ПЯТКИН)*

Рассматривается автоматическое получение сертификатов для организации протоколов https. Показан способ автоматизации процесса получения сертификатов для доменного имени. Важным аспектом процесса перевода сервера с http на https является фактор серверного окружения. В частности, подразумевается, что если на сервере установлено по apache2, то необходима перезагрузка. Текущие технологии позволяют обеспечить надежную защиту данных, в частности при использовании https.

Конфиденциальность передаваемых данных в сети является важным аспектом деятельности банковских систем или веб-приложений, хранящих персональные данные пользователя.

Утечка таких данных может нанести значительный вред как отдельным лицам, так и крупным организациям.

Для сохранения персональных данных выделено и разработано ряд мер и технологий, одной из которых является *https* протокол.

Lets Encrypt

Есть целый набор способов получить ssl-сертификат, но стоит выделить сервис Lets Encrypt, который отличает надежность и скорость получения сертификатов (за счет упразднения лишних шагов и возможности автоматизировать этот процесс), а также ряд программных средств, которые облегчают владельцу домена взаимодействие с этим сервисом.

Для автоматической выдачи сертификата конечному доменному имени используется **протокол аутентификации класса «challenge-response»** (вызов-ответ, вызов-отклик) под названием Automated Certificate Management Environment (ACME). В этом протоколе к веб-серверу, запросившему подписание сертификата, производится серия запросов для подтверждения факта владения доменом (domain validation).

Для получения запросов клиент ACME настраивает специальный TLS сервер, который опрашивается сервером ACME с применением Server Name Indication (Domain Validation using Server Name Indication, DVSNi).

С целью настройки системы обновления сертификатов на сервере, имеется ряд модулей для получения ssl сертификатов. Также важным аспектом используемого ПО является автоматическое привязывание сертификатов и перевод клиентов на 443 порт, для поддержания https.

Важным аспектом процесса перевода сервера с http на https является фактор серверного окружения. В частности, подразумевается, что если на сервере установлено по apache2, то необходима перезагрузка.

В случае работы с nginx достаточно будет использовать мягкую перезагрузку. Это приводит к мысли о том, что необходимо найти ПО для управления apache2/nginx.

Следовательно, после получения сертификата и добавления его в конфигурацию сайта необходимо автоматически перезагрузить сервер.

После этого сервер будет использовать обновленные конфигурации.

За короткое время клиент получает полностью работоспособный сайт с протоколом https.

Процесс верификации можно разделить на несколько стадий:

Создание открытого ключа по адресу domain/.well-known/acme-challenge/{challenge-token} – этот процесс необходим для подтверждения прав на домен.

Затем необходимо сделать **запрос с предоставлением данного файла сервису**.

После удачной проверки начинается **процесс получения сертификата**.

После обмена запросами имеем сертификаты в файлах fullchain.pem, cert.pem, chain.pem.

Сертификаты, как правило, выдаются на 2–3 месяца. Это необходимо учесть и добавить настройку для cron.

В основной файл конфигурации cron(/etc/crontab) необходимо добавить строку, представленную в листинге 1.

Листинг 1 – Листинг cron для Lets Encrypt

```
1: 0 0 0 */4 * /usr/local/bin/php /pathtoscript/LetsEncrypt.php
```

Таким образом, мы задали, что раз в три месяца будет обновляться сертификат.

В заключение можно сделать следующие **выводы**:

- текущие технологии позволяют обеспечить надежную защиту данных, в частности при использовании https;

- если раньше получение сертификата было сложной задачей, то при помощи сервиса Lets Encrypt эта задача перестала вызывать какие-либо проблемы у владельцев web-ресурсов.

ЛИТЕРАТУРА

1. METANIT.COM Сайт о программировании [Электронный ресурс]. – Режим доступа: <https://metanit.com/nosql/mongodb/1.1.php>. – Дата доступа: 18.09.2017.
2. Руководство по MongoDB. Моделирование данных [Электронный ресурс]. – Режим доступа: http://proselyte.net/tutorials/mongodb/data_modeling/. – Дата доступа: 17.09.2017.
3. Википедия : свободная энцикл. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/JSON/>. – Дата доступа: 21.09.2017.
4. W3II Latest web development tutorials [Электронный ресурс]. – Режим доступа: http://www.w3ii.com/ru/mongodb/mongodb_advantages.html. – Дата доступа: 23.09.2017.

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.[822+823]

СЕМАНТИЧЕСКИЕ ФРЕЙМЫ: КЛАССИФИКАТОРЫ И КВАЛИФИКАТОРЫ

А.И. ДУНЧЕНКО

(Представлено: Д.В. ПЯТКИН)

Рассматривается концепция представления знаний в интеллектуальных компьютерных системах. Раскрывается сущность новой модели, основанной на семантических фреймах и ее связь с уже существующими решениями. Описываются отличительные особенности – классификаторы и квалификаторы, которые являются дополнениями к базовой фреймовой модели.

Введение. При создании систем искусственного интеллекта ключевым вопросом является выбор способа представления и обработки информации об объектах и явлениях реального мира применительно к возможностям современных ЭВМ. В зависимости от типа решаемой задачи, существует два подхода к этой проблеме. *Первый* – это использование искусственных нейронных сетей (ИНС) [1]. ИНС – это аппаратно или программно реализованная сеть, работа которой основана на тех же принципах, что и мозг высших представителей фауны. Каждая такая ИНС получается путем комбинирования множества различных нейронов, реализующих ту или иную математическую функцию. Недостатком нейронных сетей является то, что они требуют длительного обучения и являются ресурсоемким решением. Второй подход заключается в *применении специфической нотации*. Традиционными представителями данной группы являются: *правила* (выражения IF-THEN-ELSE), *предикаты* [2], *семантические сети* [3], *фреймы* [4].

Ни одно из перечисленных решений не является универсальным: все они обладают как определенными преимуществами друг перед другом, так и некоторыми недостатками. Поэтому выбор той или иной нотации сильно зависит от конкретно поставленной перед разработчиками задачи.

Семантические фреймы (СФ) – это попытка создания новой модели представления знаний, в основе которой лежит все та же концепция фреймов, только дополненная элементами из семантических сетей, предикатов и нечеткой логики.

В модели СФ отдельно взятый слот или даже целый фрейм может выражать *семантическое отношение* между связанными элементами. Для этой цели служат *классификаторы*.

Классификаторы

Классификатор – это символическая константа (в текущей спецификации это так; в будущих версиях вполне возможно, что константы будут заменены на классы, методы которых будут определять правила управления отношениями), выражающее характеристику своего операнда в наиболее обобщенной форме. То есть при определении перечня стандартных классификаторов делался упор на минимализм: из ряда близких по смыслу терминов выбирался тот, который имеет более абстрактное значение, включающее в себя множество смежных понятий. Каждый слот или фрейм может иметь несколько не противоречащих друг другу классификаторов, правила, назначения которых определяются спецификацией и приведены в таблицах 1–7.

Таблица 1. – Классификаторы фреймов

Классификатор	Описание
<i>set</i>	Коллекция связанных сущностей: сфера, группа, пространство имен, пакет и т.п.
<i>class</i>	То же, что и в ООП.
<i>prototype</i>	То же, что и в ООП.
<i>enumeration</i>	Перечисление символических констант (аналогично <i>enum</i> в С). Например – <i>сезон: зима, весна, лето, осень</i> .
<i>relation</i>	Множество тесно связанных сущностей. Слоты содержат объекты, связанные данным отношением.
<i>activity</i>	Используется для описания процедурных знаний. Слоты содержат участников (затрагиваемые объекты) и характеристики деятельности (события, явления и т.п.).
<i>object</i>	Экземпляр некоторого класса/прототипа или анонимный объект без определенного типа.

Таблица 2. – Модификаторы

Классификатор	Описание
<i>fixed</i>	Не позволяет добавлять/удалять слот из фрейма или изменять значение слота.
<i>not-inherited</i>	Фреймы, помеченные данным классификатором, не могут наследоваться (<i>class, enumeration</i>). Слоты с таким классификатором не наследуются.
<i>shared</i>	Экземпляры класса вместо копии получают ссылку на слот.
<i>attribute</i>	Анонимный слот. Значение слота содержит некоторую обобщенную характеристику (прилагательное, наречие, числительное).

Таблица 3. – Классификаторы слотов *set*-фреймов

Классификатор	Описание
<i>includes</i>	Безымянный слот, содержащий ссылку на вложенное множество.
	Слоты без перечисленных выше классификаторов содержат обычные элементы множества.

Таблица 4. – Классификаторы слотов *class*-фреймов

Классификатор	Описание
<i>extends</i>	Безымянная ссылка на базовый (расширяемый) класс.
<i>includes</i>	Безымянная ссылка на вложенный класс.
<i>activity</i>	Слот описывает деятельность, производимую экземпляром класса, или событие, на которое он может реагировать.
	Слоты без перечисленных выше классификаторов содержат поля, копии которых будут назначены всем экземплярам класса.

Таблица 5 – Классификаторы слотов *enumeration*-фреймов

Классификатор	Описание
<i>extends</i>	Безымянная ссылка на базовое (расширяемое) перечисление.
<i>activity</i>	Слот описывает деятельность, производимую элементом перечисления, или событие, на которое он может реагировать.
	Слоты без перечисленных выше классификаторов содержат элементы перечисления, а их имена являются идентификаторами этих значений.

Таблица 6. – Классификаторы слотов *relation*-фреймов

Классификатор	Описание
<i>extends</i>	Безымянная ссылка на базовое (расширяемое) отношение.
<i>activity</i>	Слот описывает деятельность, подразумеваемую данным отношением, или событие, которое может наступить в определенный момент.
	Слоты без перечисленных выше классификаторов содержат элементы, связанные отношением (являющиеся активными или пассивными участниками).

Таблица 7. – Классификаторы слотов *activity*-фреймов

Классификатор	Описание
<i>in</i>	Входной параметр (аргумент).
<i>out</i>	Выходной параметр (результат).
<i>in activity</i>	Деятельность, которая может инициировать данную активность (причина).
<i>out activity</i>	Деятельность, которая может быть инициирована данной активностью (последствие).
<i>includes</i>	Деятельность, на которой основана текущая активность.
	Слоты без перечисленных выше классификаторов содержат объекты, которые подвергаются воздействию или являются участниками деятельности.

Необходимо отметить, что слоты, помеченные классификатором *extends*, всегда располагаются в первой секции фрейма (т.е. в самом начале списка); за ними могут следовать *includes*-слоты, а уже затем все остальные. Исключением являются *activity*-фреймы – для них порядок следующий: входные параметры, выходные параметры, *includes*-слоты, прочие слоты. У экземпляров классов и прототипов первый слот должен иметь классификатор *class* или *prototype* и ссылаться на соответствующий объект.

Квалификаторы

Квалификатор – это предикат, записанный в форме *F-выражения*. В модели СФ квалификаторы служат для внесения ограничений, проверки соответствия условиям и т.п. Следует сразу указать, что система представления знаний с помощью СФ все еще находится на стадии разработки и апробации, поэтому в текущей спецификации потенциал *F-выражений* реализуется не полностью. На данный момент квалификаторы могут использоваться в качестве триггеров для проверки значений, присваиваемых слотам.

F-выражение (от англ. *F-expression* – functional expression) представляет собой запись, идентичную математической нотации определения *функции* [5], которая в форме *EBNF* [6] выглядит так:

```
f-expression = identifier, '(', arguments, ')';
arguments    = [ argument, { ',', argument } ];
argument     = value;
array        = '[', [ value, { ',', value } ], ']';
value        = f-expression | identifier | constant | array.
```

Причины, по которым был выбран такой синтаксис:

- 1) *F-выражения* по своей природе подобны функциям и не должны иметь побочных эффектов;
- 2) данный синтаксис, в отличие от *M-выражений* [7], используется для объявления/вызова функции во многих языках программирования (ЯП), производных от Algol, и, поэтому, является более привычным для восприятия;
- 3) данный синтаксис, в отличие от *символических выражений* [8], не привязан к конкретному ЯП (Lisp) и, используя *регулярные выражения* [9], может быть легко сконвертирован в AST [10], построенное на базе простых списков.

В отличие от обычных функций, *F-выражения* получают свои параметры в исходной форме и могут вычислять их по мере необходимости. Это позволяет унифицировать понятия *макрос* и *функция*.

Так, в зависимости от значения первого аргумента, форма *if(..., ..., ...)* вычисляет и возвращает либо второй, либо третий параметр.

Каждый квалификатор, будучи предикатом, по сути, должен возвращать строгое *логическое значение* [11]: $P() \in \{ \perp, T \}$.

Следует заметить, что в концепции *F-выражений* используется подобие *троичной логики* [12], где значения $(-1, 0, 1)$ представлены соответственно константами **\$f**, **\$?** и **\$t**.

Результаты логических операций над ними представлены в таблице 8.

Таблица 8. – Логические операции в троичной логике

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \oplus B$	$A \equiv B$
\$f	\$f	\$t	\$f	\$f	\$t	\$f	\$t
\$f	\$?	\$t	\$f	\$?	\$t	\$?	\$?
\$f	\$t	\$t	\$f	\$t	\$t	\$t	\$f
\$?	\$f	\$?	\$f	\$?	\$?	\$?	\$?
\$?	\$?	\$?	\$?	\$?	\$?	\$?	\$?
\$?	\$t	\$?	\$?	\$t	\$t	\$?	\$?
\$t	\$f	\$f	\$f	\$t	\$f	\$t	\$f
\$t	\$?	\$f	\$?	\$t	\$?	\$?	\$?
\$t	\$t	\$f	\$t	\$t	\$t	\$f	\$t

Важным дополнением является то, что константа **\$?** (undefined) представляет собой *суперзначение*, символизирующее отсутствие конкретных знаний как таковых, поэтому она может фигурировать в выражениях, оперирующих различными типами, например:

$\text{sum}(1,2,3, \$?) = \$?; \text{union}([1, 2, 3], [4, 5, 6], \$?) = \$?.$

По-умолчанию, значения слотов инициализируются данной константой.

Спецификация СФ в новой редакции включает следующие F-выражения:

- 1) математические операции: *neg, add, sub, mul, div, mod, rem, pow, exp, log, root2, root3*;
- 2) операторы сравнения: *below, below_or_equals, above, above_or_equals, equals, is_positive, is_negative*;
- 3) логические конструкции: *and, or, xor, not*;
- 4) предикаты для проверки типов: *is_number, is_string, is_array, is_boolean, is_strict_boolean, is_undefined*;
- 5) операции над контейнерами: *array, size, nth, is_empty, has*;
- 6) операции над множествами: *union, intersection, difference, symmetric_difference, contains*;
- 7) условные конструкции: *if*.

Заключение

Модель СФ претендует на статус универсальной системы представления знаний, включающей в себя свойства и возможности других наиболее распространенных нотаций. Насколько данное решение окажется приемлемым для разработчиков программного обеспечения, получив необходимую поддержку и стимул к дальнейшему развитию, зависит от множества факторов, в частности:

- 1) всесторонней продуманности модели, гибкости ее архитектуры и способности к расширению;
- 2) наличия исчерпывающей документации;
- 3) возможности и заинтересованности в стандартизации;
- 4) наличия достаточного количества примеров успешного применения и внедрения.

Если проанализировать текущую ситуацию на рынке коммерческого и в секторе свободного программного обеспечения, то можно заметить, что преобладающим является стремление к in-house разработке. Каждый год появляются новые бесполезные ЯП, форматы и протоколы, дистрибутивы операционных систем, библиотеки и фрейворки, дублирующие функциональность уже существующих решений. Это происходит потому, что особо консервативно настроенные компании, подверженные синдрому NIH (Not Invented Here [13]) предпочитают разрабатывать самостоятельно не только бизнес-логику, но и все сопутствующие технологии, тем самым пытаясь снизить риски, повысить конкурентно-способность и удержать финансовый приток от дочерних и партнерских фирм, использующих их решения. Что касается open-source сообщества, то оно страдает от так называемого «holy wars», неспособности к принятию единогогласных решений и организованной разработке.

Таким образом, в условиях текущей ситуации на рынке коммерческого и в секторе свободного программного обеспечения маловероятно, что система представления знаний, основанная на СФ, получит должное внимание и тем более широкое распространение.

ЛИТЕРАТУРА

1. Artificial neural network [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Artificial_neural_network. – Date of access: 20.09.2017.
2. First-order logic [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/First-order_logic. – Date of access: 20.09.2017.
3. Semantic network [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Semantic_network. – Date of access: 20.09.2017.
4. Frame (artificial intelligence) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/Frame_\(artificial_intelligence\)](https://en.wikipedia.org/wiki/Frame_(artificial_intelligence)). – Date of access: 20.09.2017.
5. Function (mathematics) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/Function_\(mathematics\)](https://en.wikipedia.org/wiki/Function_(mathematics)). – Date of access: 24.09.2017.
6. Extended Backus–Naur form [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Extended_Backus–Naur_form. – Date of access: 24.09.2017.
7. M-expression [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/M-expression>. – Date of access: 24.09.2017.
8. S-expression [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/S-expression>. – Date of access: 24.09.2017.
9. Regular expression [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Regular_expression. – Date of access: 24.09.2017.
10. Abstract syntax tree [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Abstract_syntax_tree. – Date of access: 24.09.2017.
11. Boolean algebra [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Boolean_algebra. – Date of access: 25.09.2017.
12. Three-valued logic [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Three-valued_logic. – Date of access: 25.09.2017.
13. Not invented here [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Not_invented_here. – Date of access: 25.09.2017.

УДК 004.[822+823]

**СЕМАНТИЧЕСКИЕ ФРЕЙМЫ:
РАЗРАБОТКА ФОРМАТА СЕРИАЛИЗАЦИИ ДАННЫХ****А.И. ДУНЧЕНКО****(Представлено: Д.В. ПЯТКИН)**

Представлено понятие о семантических фреймах. Сделан вывод о необходимости разработки формата сериализации для этой модели представления знаний, перечисляется перечень проблем, которые при этом требуют решения. Выполнен анализ наиболее очевидных способов достижения цели. Показано, как существующие наработки и стандарты могут быть адаптированы для решения поставленной задачи.

Семантические фреймы (СФ) – это универсальная система представления знаний об окружающем нас мире средствами ЭВМ, которая позволяет интерпретировать и структурировать поступающую информацию в виде, приемлемом для понимания как человеком (как правило, экспертом в конкретной проблемной области), так и компьютерной системой, выполняющей обработку этой информации.

Семантические фреймы являются гибридной моделью, базирующейся на концепциях *семантических сетей* [1] и *фреймов* [2] (отсюда и соответствующее название) и дополняющей их элементами *логики первого порядка* [3] и *нечеткой логики* [4].

При проектировании любого программного обеспечения, независимо от его сложности и того, выполняет ли оно конвейерную обработку данных или глобально оперирует средой выполнения (имеет побочные эффекты), всегда возникает вопрос о необходимости введения фиксированных механизмов обмена данными – протоколов взаимодействия – между компонентами системы: функциями, объектами, модулями или даже отдельными приложениями [5]. Примерами такого взаимодействия являются: *FFI* [6], *CGI* [7], *FIFO* [8], *сокеты* [9], *каналы* [10], сообщения.

В связи с этим рассмотрению подлежат следующие проблемы:

1) *обеспечение аппаратной независимости*, которое столь же актуально и в настоящее время, когда большинство настольных пользовательских систем работает на little-endian i386- и amd64-платформах, мобильные устройства – на dual-endian ARM-процессорах, а в сети Internet стандартным считается big-endian порядок байтов;

2) *обеспечение независимости от типа операционной и файловой систем*, так как даже при использовании виртуальных машин и интерпретируемых высокоуровневых языков программирования, по-прежнему имеют место трудности в реализации, обусловленные кодировками, регистро-(не)зависимостью, механизмами работы с файлами, устройствами и т.п.;

3) *обеспечение независимости от применяемого языка программирования (ЯП) или версии компилятора* (наиболее ярко выражена эта проблема в open-source сообществе, где разработчики имеют негативную тенденцию к внесению изменений в ABI [11] или API [12] с выходом каждой новой мажорной версией своего ПО);

4) *in-house разработка или применение уже существующего стандарта* (последнее, как правило, является более предпочтительным решением).

Модель СФ представляет собой не только спецификацию, но и фактическую реализацию – библиотеку, написанную на ЯП Common Lisp, – которая предназначена для работы в составе более сложной интеллектуальной программной системы. Поэтому для нее также справедливы все перечисленные ранее проблемы, ведь конечное ПО в процессе своего функционирования предполагает:

- 1) сохранение и чтение файлов с жесткого диска;
- 2) обмен данными между компонентами посредством сокетов;
- 3) отображение информации пользователю.

Таким образом, необходимо разработать формат для сериализации данных, который позволит не только эффективно выполнять операции ввода/вывода, но и будет приемлемым для восприятия человеком (экспертом или программистом, выполняющим проверку/отладку системы). Следуя этим требованиям, которые, кстати, соответствуют Unix-философии, целесообразно применить текстовый формат. При этом саму проблему можно разделить на две подзадачи:

1) там, где предполагается оперирование данными только в рамках библиотеки – сохранение и чтение из файлов – использовать *символические выражения* [13], на которых, собственно, и базируется синтаксис Common Lisp;

2) в случае если необходимо передавать данные между программами посредством CGI или сокетов (например, между сервером базы знаний и веб-сервером), утилизировать один из существующих текстовых языков разметки [14], адаптировав его под конкретные нужды.

Среди всех существующих языков разметки и представления данных наиболее подходящими являются следующие кандидаты: JSON, YAML, XML.

Рассмотрим их более подробно.

JSON (JavaScript Object Notation [15]) является подмножеством синтаксиса языка JavaScript. Считается, что, несмотря на свое происхождение, JSON 1) не зависит от JavaScript и может использоваться с любым другим языком программирования и 2) за счет своей лаконичности по сравнению с XML, может быть более подходящим для сериализации сложных структур. Однако при детальном рассмотрении оказывается, что оба эти утверждения далеки от истины.

Во-первых, система типов JSON полностью соответствует таковой именно языка JavaScript, поэтому, в сравнении с большинством других ЯП, зачастую либо оказывается крайне примитивной, либо не позволяет выполнить однозначную конверсию между значениями. Примеры: 1) в Common Lisp нет различий между константами *null* и *false*, оба значения представлены как *nil*, в Scheme вместо *nil* вообще используется пустой список; 2) символы и комплексные числа (C, Common Lisp, Scheme, Ruby, Python) приходится представлять в виде строки, в результате чего выполнить обратную интерпретацию в ряде случаев невозможно (что есть "4" – строка, символ или число?); 3) объекты JSON не являются полноценными hash-таблицами – в отличие от, к примеру, C++, Ruby или Python, ключами могут быть только строки.

Во-вторых, простой по сравнению с тем же XML синтаксис в то же время является и недостатком, так как провоцирует избыточные вложенные структуры там, где в XML можно обойтись лишь атрибутами. Кроме того, JSON просто-напросто не умеет представлять рекурсивные структуры.

YAML (YAML Ain't Markup Language [16]) ставит перед собой следующие задачи:

1) быть выразительным, расширяемым и доступным в понимании человеку;
2) поддерживать структуры данных, характерные для различных ЯП, обеспечивая легкую переносимость;

3) быть простым в реализации и использовании за счет обработки в один проход.

YAML имеет больше типов данных, чем JSON, поддерживает теги и ссылки на блоки. И тем не менее этот формат также не желателен в качестве решения для поставленной задачи. Создатели формата, как и автор языка Python, повторили ту же ошибку, сделав пробельные символы синтаксическим элементом. Это часто является причиной трудно обнаруживаемых ошибок и приводит к значительным трудностям как при визуальном просмотре многостраничных текстов, так и при написании утилит, предназначенных для анализа (обработки) документов. Иначе говоря, YAML является приемлемым вариантом, если речь идет о работе с файлами конфигурации или небольшой базой данных, но для представления таких сложных структур, как фреймы, этот формат не подходит.

Что касается XML (eXtensible Markup Language [17]), то несмотря на предостаточное количество критики в адрес данного формата (например, что для обработки документов требуется больше оперативной памяти и вычислительных ресурсов, чем в случае с JSON/YAML), он является если не идеальным, то вполне обоснованным решением проблемы: гибкий синтаксис XML позволяет 1) адаптировать его под любые задачи, связанные с разметкой и 2) сериализовать сколь угодно сложные иерархические структуры данных. Таким образом, взамен потери производительности мы получаем необычайно мощное средство представления данных, уступающее по своим возможностям только S-выражениям.

Определившись с форматом можно приступить к этапу проектирования структуры документа, т. е. опираясь на иерархию классов, предназначенных для описания и работы с фреймами, определить перечень используемых тегов, их атрибутов, а также правила формирования документа. Для этого можно воспользоваться формальной системой для определения синтаксиса EBNF (extended Backus-Naur form [18]):

document	= prologue, pool node;
prologue	= '<?xml version="1.0" encoding="UTF-8" ?>';
pool node	= '<pool', [' name="", pool name, ""], '>', { frame node }, '</pool>';
frame node	= '<frame id="", frame id, ""', [' name="", frame name, ""], '>', frame head node, frame body node, '</frame>';
frame head node	= '<head>', [classifiers node], [clarifiers node], '</head>';
classifiers node	= '<classifiers>', { classifier node }, '</classifiers>';
classifier node	= '<c>', classifier, '</c>';
clarifiers node	= '<clarifiers>', { clarifier node }, '</clarifiers>';
clarifier node	= ' ', predicate, ' ';
frame body node	= '<body>', { slot node slot reference node }, '</body>';

slot node	= '<slot id=" ', slot id, ' " ', [' name = " ', slot name, ' " '], '>', slot head node, slot data node, '</slot>';
slot head node	= '<head>', [classifiers node], [clarifiers node], [membership node], '</head>';
membership node	= '<membership>', membership, '</membership>';
membership	= <i>real number between 0 and 1</i> ;
slot data node	= '<data type=" ', slot data type, ' ">', slot data, '</data>';
slot data type	= 'boolean' 'character' 'number' 'string' 'JSON' 'S-exp', 'regex', 'base16' 'base64' 'slot-ref' 'frame-ref' custom data type;
custom data type	= <i>string</i> ;
slot reference node	= '<slot-ref id=" ', slot id, ' " />';
frame id	= id;
slot id	= id;
id	= <i>string</i> .

Как видно из правил, если пул, фрейм или слот не имеет идентификатора, атрибут *name* следует опускать. Для классификаторов и квалификаторов, которые с большой долей вероятности будут самыми часто встречающимися элементами, используются краткие теги **<c>** и **<p>** (сокращения от *classifier* и *predicate* соответственно). Сами предикаты вписываются в виде исходных *F-выражений* (F-expressions), а не в полученной после парсинга символической форме. Если слот принадлежит фрейму, то используется элемент **<slot>**, иначе (например, если фрейм ссылается на слот с классификатором *shared*) – **<slot-ref>**.

Теперь, когда общая структура документа четко определена, следует акцентировать внимание на промежуточном представлении СФ с помощью символических выражений, т. е. выбрать такой способ организации данных, который позволит с легкостью не только генерировать XML-разметку, но и выполнять обратную операцию – строить фреймы на основе данной формы.

Эта задача легко решается, если для работы с XML использовать пакет *cl-xmles*, который, кстати, и использует S-выражения в качестве входных/выходных данных. Построение каждого элемента починено весьма простому правилу, которое в нотации EBNF выглядит следующим образом:

node	= '(', node name, node attributes, node children, ')';
node name	= <i>literal</i> ;
node attributes	= '(', { attribute }, ')';
attribute	= '(', attribute name, attribute value, ')';
attribute name	= <i>literal</i> ;
attribute value	= <i>value</i> ;
node children	= <i>value</i> { node };
literal	= <i>string</i> <i>symbol</i> ;
value	= <i>any Lisp object but pair</i> .

Такой незаурядный способ организации данных позволяет эффективно выполнять их обработку, применяя функциональный подход к программированию и встроенные в стандартную библиотеку языка Common Lisp макросы и функции. Более того, учитывая тот факт, что *cl-xmles* автоматически конвертирует любые значения в строки, можно существенно сэкономить память, если в качестве имен тегов использовать символы из пакета *keywords*.

Подводя итоги, сделаем вывод, что поставленная задача по разработке формата для сериализации СФ успешно решена: для внутренних нужд библиотеки, реализующей спецификацию СФ, будет использоваться промежуточное представление, основанное на символических выражениях, а для межпроцессного взаимодействия, где требуется обеспечить максимальную изолированность компонентов и независимость от программных решений, – язык разметки XML.

Единственной нерешенной проблемой остается сериализация триггеров. Причины, по которым это на данный момент представляется невозможным, следующие:

- 1) при сохранении замыканий нужно также сохранять информацию о среде выполнения, а в случае с Common Lisp, контекст может быть как статическим, так и динамическим;
- 2) триггеры могут быть написаны на разных ЯП, не обязательно Common Lisp;

3) стек может и вовсе содержать цепочку вызовов из разных ЯП, например:
 $C \rightarrow \text{Common Lisp} \rightarrow C \rightarrow \text{Tcl} \rightarrow C \rightarrow \text{Common Lisp} \rightarrow C$,
в результате чего выполнить полный захват и последующую корректную инициализацию невозможно в принципе.

ЛИТЕРАТУРА

1. Semantic network [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Semantic_network. – Date of access: 20.09.2017.
2. Frame (artificial intelligence) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/Frame_\(artificial_intelligence\)](https://en.wikipedia.org/wiki/Frame_(artificial_intelligence)). – Date of access: 20.09.2017.
3. First-order logic [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/First-order_logic. – Date of access: 20.09.2017.
4. Fuzzy logic [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Fuzzy_logic. – Date of access: 20.09.2017.
5. Inter-process communication [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Inter-process_communication. – Date of access: 20.09.2017.
6. Foreign function interface [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Foreign_function_interface. – Date of access: 20.09.2017.
7. Common Gateway Interface [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Common_Gateway_Interface. – Date of access: 20.09.2017.
8. FIFO (computing and electronics) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/FIFO_\(computing_and_electronics\)](https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics)). – Date of access: 20.09.2017.
9. Unix domain socket [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Unix_domain_socket. – Date of access: 20.09.2017.
10. Pipeline (computing) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/Pipeline_\(computing\)](https://en.wikipedia.org/wiki/Pipeline_(computing)). – Date of access: 20.09.2017.
11. Application binary interface [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Application_binary_interface. – Date of access: 20.09.2017.
12. Application programming interface [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Application_programming_interface. – Date of access: 20.09.2017.
13. S-expression [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/S-expression>. – Date of access: 21.09.2017.
14. Markup language [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Markup_language. – Date of access: 21.09.2017.
15. JSON [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/JSON>. – Date of access: 21.09.2017.
16. YAML [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/YAML>. – Date of access: 21.09.2017.
17. XML [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/XML>. – Date of access: 21.09.2017.
18. Extended Backus–Naur form [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Extended_Backus%E2%80%93Naur_form. – Date of access: 21.09.2017.

УДК 004.65

**ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ
ВЕБ-ПРИЛОЖЕНИЯ «АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА
ВЕДЕНИЯ ОТЧЕТНОСТИ ДЛЯ ОАО «НАФТАН»»****А.Н. РОДЧЕНКОВ***(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)*

Представлен объект разработки – веб-приложение «Автоматизированная информационная система ведения отчетности для ОАО «Нафтан»». Рассмотрены вопросы проектирования базы данных для информационной системы ведения отчетности.

Цель работы – создание веб-приложения «Автоматизированная информационная система ведения отчетности для ОАО «Нафтан»». Для создания приложения, работающего с базой данных, необходимо сначала спроектировать саму базу. Для удобства последующей разработки проектируемая база должна быть максимально проста для понимания и нормализована. Для проектирования базы данных необходимо выявить необходимый набор сущностей, указать необходимые атрибуты, характеризующие каждую сущность, а также определить связи между сущностями.

На предприятии имеется в наличии различное измерительное оборудование, которое подлежит складскому учету. При учете имеющегося оборудования указывается заглавие секции, на балансе которой оно находится. Оборудование делится на два типа: клапаны и приборы. Исходя из этого получаем следующий набор сущностей:

- 1 «Клапаны».
- 2 «Приборы».
- 3 «Секции».
- 4 «Склад».

Связь между сущностями «Клапаны», «Приборы», «Секции» и «Склад» представлена на рисунке 1.

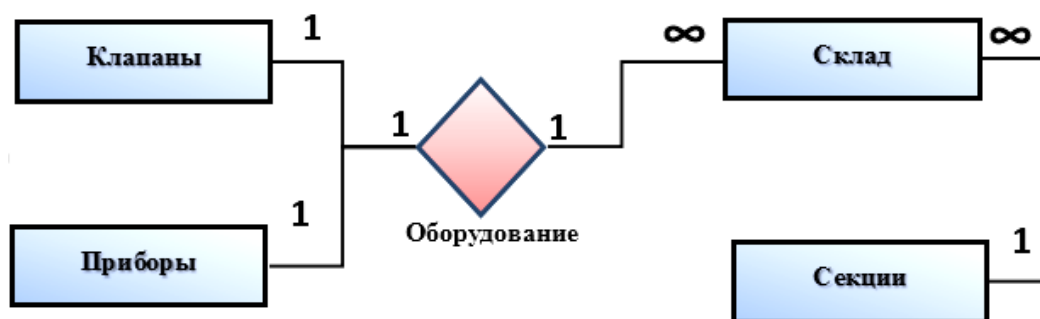


Рисунок 1. – Связь между сущностями «Клапаны», «Приборы», «Секции» и «Склад»

Также имеются различные технические позиции, за которыми, в свою очередь, закрепляются позиции размещения измерительных клапанов. На данных позициях размещается различное измерительное оборудование, которое есть на складе. Для фиксации закреплённого складского оборудования на позициях необходимо определить такую сущность, как «Закрепление». Для более детального представления о закреплении различного рода оборудования на технических позиция и позициях измерительных клапанов введем дополнительные сущности – «Закрепление приборов» и «Закрепление клапанов».

Исходя из набора вышеописанных фактов список выявленных сущностей будет выглядеть следующим образом:

- 1 «Тех. Позиция».
- 2 «Позиция ИКС».
- 3 «Закрепление Клапанов».
- 4 «Закрепление Приборов».

Связь между сущностями «Тех. Позиция», «Позиция ИКС», «Закрепление Клапанов» и «Закрепление Приборов» представлена на рисунке 2.

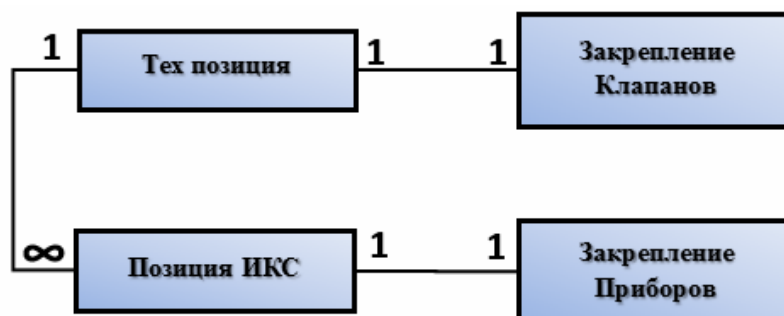


Рисунок 2. – Связь между сущностями «Тех. Позиция», «Позиция ИКС», «Закрепление Клапанов» и «Закрепление Приборов»

Для построения схемы базы данных необходимо определить совокупность отношений между сущностями, которые будут использоваться в базе. Также важно, чтобы в отношениях между сущностями выполнялись условие целостности сущностей и условие целостности ссылок.

Условие целостности сущностей заключается в следующем: каждый кортеж любого отношения должен отличаться от любого другого кортежа этого отношения, то есть любое отношение должно обладать первичным ключом, а условие целостности ссылок, в свою очередь, заключается в том, что для каждого значения внешнего ключа, появляющегося в дочернем отношении, в родительском отношении должен найтись кортеж с таким же значением первичного ключа [3].

Схема спроектированной базы данных представлена на рисунке 3.

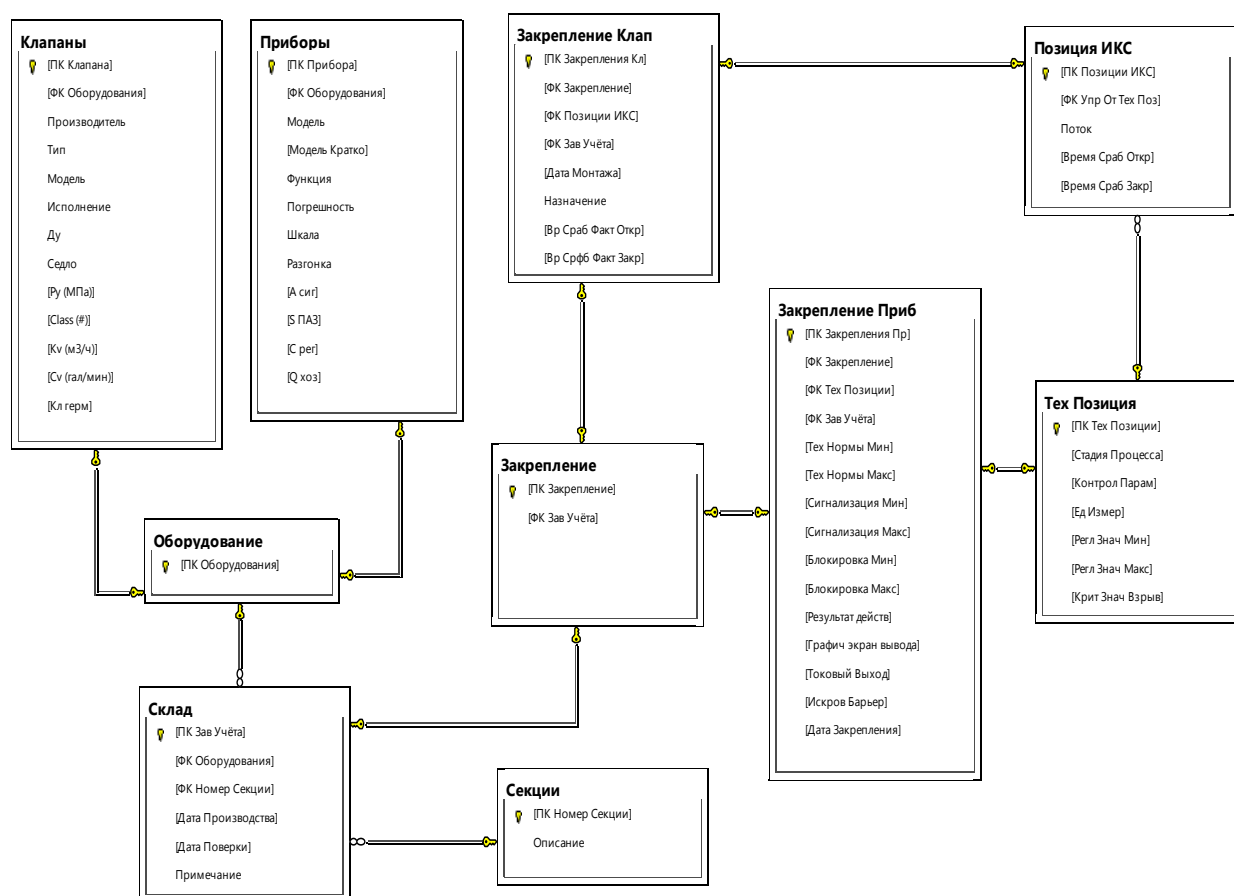


Рисунок 3. – Набор необходимых отношений базы данных

Выбор СУБД является сложной задачей и должен основываться, в первую очередь, на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид

программного продукта и категория пользователей (или профессиональные программисты, или конечные пользователи, или и то, и другое).

Другими показателями, влияющими на выбор СУБД, являются [2]:

- удобство и простота использования;
- качество средств разработки, защиты и контроля базы данных;
- уровень коммуникационных средств в случае применения ее в сетях;
- фирма-разработчик;
- стоимость.

Среди всего множества доступных СУБД выбор был сделан в пользу MS SQL Server 2014. Выбор средств решения поставленной задачи исходил из того, какое программное и аппаратное обеспечение используется в данный момент на предприятии.

Система MS SQL Server 2014 позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio. MS SQL Server обеспечивает высочайший уровень безопасности, надежности и масштабируемости для критически важных приложений. Чтобы использовать новые возможности, постоянно возникающие в быстро меняющемся деловом мире, предприятиям нужно быть способными быстро создавать и развертывать решения, управляемые данными. MS SQL Server 2014 позволяет сократить затраты времени и средства, требуемые на управление и развертывание таких приложений. Следует учесть, что фирма-разработчик данной СУБД является также разработчиком самой распространенной ОС. В финансовом плане важным фактором является то, что существуют бесплатные сборки данной СУБД.

Для правильной работы и сохранения ссылочной целостности для сущностей в базе были разработаны триггеры на добавление/удаление/редактирование, которые предотвращают добавление в базу одинаковых записей и организуют каскадное удаление, не оставляя неактуальные ссылки.

Заключение

В результате проделанной работы спроектирована база данных, при помощи которой можно создать автоматизированную информационную систему, направленную на сокращение временных затрат работников предприятия, занимающегося ведением различной внутренней документации и автоматизировать формирование отчетов.

ЛИТЕРАТУРА

1. Малыгина, М. Базы данных: основы, проектирование, использование / М. Малыгина. – М. : БНУ, 2004. – 512 с.
2. Статья «Выбор СУБД» [Электронный ресурс]. – Режим доступа <http://websideworld.com/main/data-storage/6/2>. – Дата доступа: 27.09.2017.
3. Зависимости по соединению и пятая НФ (5NF). [Электронный ресурс]. – Режим доступа: <http://sdamzavas.net/3-6246.html>. – Дата доступа: 27.09.2017.

УДК 004.031.042

**ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНОЙ ЧАСТИ
ВЕБ-ПРИЛОЖЕНИЯ «АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА
ВЕДЕНИЯ ОТЧЕТНОСТИ ДЛЯ ОАО «НАФТАН»»**

А.Н. РОДЧЕНКОВ

(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

Представлен объект разработки – веб-приложение «Автоматизированная информационная система ведения отчетности для ОАО «Нафтан»». Рассмотрены вопросы проектирования функциональной части веб-приложения «Автоматизированная информационная система ведения отчетности для ОАО «Нафтан»».

Создание программного обеспечения для надежного хранения данных о функционировании предприятия и формирование внутренних отчетов является целью данного исследования. Основная цель разработки приложения – сделать хранение данных о функционировании предприятия и формирование отчетов как можно больше простым и универсальным.

Первым этапом проектирования программного обеспечения является *определение функциональных и других возможностей, которым программное обеспечение должно соответствовать*. Предполагается, что с данной информационной системой будут работать сотрудники установки, и они должны иметь возможность решать с ее помощью следующих задач:

- ввод и редактирование данных о технологических позициях, приборах и закреплениях в удобной форме;
- просмотр данных о технологических позициях, приборах и закреплениях в отсортированном виде;
- просмотр и подготовка к выводу списков технологических позиций и закрепленных за ними приборов;
- формирование отчетов согласно стандартам технического регламента;
- экспорт информации и сформированных отчетов из базы данных в Excel и наоборот;
- организация надежного хранения данных;
- организация поиска по ID или части ID.

Таким образом, при разработке продукта были выдвинуты следующие требования к функциональным характеристикам:

- предоставлять пользователю информацию, хранящуюся в информационной системе;
- предоставлять базовый функционал сортировки информации при работе с информационной системой;
- предоставлять пользователю возможность вносить изменения в информацию, хранящуюся в информационной системе;
- предоставлять пользователю возможность экспорта информации из базы данных в Excel и наоборот;
- предоставлять авторизованному пользователю возможность формирования отчетов согласно внутреннему техническому регламенту.

Так как разрабатываемое программное средство является автоматизированной информационной системой, то требования, предъявляемые к любой автоматизированной информационной системе, будут распространены и на разрабатываемый программный продукт. Среди них выделяют:

1) наличие визуальной среды для просмотра и обработки информации. Для обработки информации, которая поступает в информационную систему, необходимо разработать удобную оболочку, через которую, посылая запросы к модулям логики работы системы, пользователь осуществляет необходимую ему обработку данных. Для такой визуальной среды важными характеристиками являются дизайн пользовательского интерфейса, интуитивность и удобство навигации;

2) возможность выполнять с информацией, поступающей в информационную систему, различные стандартные операции. К таким операциям относят операции добавления, удаления, а также изменения (редактирования) данных;

3) логическое разделение автоматизированной информационной системы на программные компоненты. В данном случае информационную систему удобнее всего разделить на четыре основных вышеперечисленных компонента: база данных, подсистема генерации отчетов, подсистема импорта данных из Excel, редактор базы данных;

4) ориентированность на пользователя – для работы с разрабатываемым программным средством не требуется предварительной подготовки персонала;

5) стабильность работы – приложение должно обеспечивать возможность корректного завершения в случае ошибки.

Следующим этапом проектирования является *разработка функциональной структуры программного обеспечения*. В данном случае информационную систему удобнее всего разделить на четыре основных компонента: база данных, подсистема генерации отчетов, подсистема импорта данных из Excel, редактор базы данных.

Среди функций разрабатываемых подсистем можно выделить следующее:

- 1 Для подсистемы база данных:
 - удобное и надежное хранение необходимой информации;
 - обеспечение целостности хранимой информации.
- 2 Для подсистемы импорта данных из Excel:
 - непосредственно импорт данных из Excel в базу данных;
 - возможность дополнения или обновления имеющийся базы с помощью импорта данных из Excel;
 - возможность выбора заполняемой таблицы;
 - возможность выбора файла для импорта данных из Excel.
- 3 Для подсистемы генерации отчетов:
 - генерация отчетов различных типов;
 - возможность просмотра генерируемых отчетов;
 - изменение параметров генерации отчетов;
 - возможность сохранения отчетов в различных форматах.
- 4 Для подсистемы редактор базы данных:
 - добавление записей в базу данных согласно формату;
 - удаление записей из базы данных;
 - изменение записей в базе данных по ключу;
 - различные варианты группировок при работе с базой данных для более удобной обработки информации;
 - возможность ускоренного поиска по части слова в выбранной ячейке;
 - возможность скрыть меню, расширение рабочей области;
 - ограничение количества выводимых полей на одной странице.

Следующим этапом проектирования программного обеспечения является *создание диаграммы вариантов использования* [1].

При создании диаграмм вариантов использования нужно выделить два вида сущностей – актеры и прецеденты. Актер – это множество логически связанных ролей, исполняемых при взаимодействии с прецедентами. Прецедент – это описание множества последовательных событий, включая варианты, выполняемые системой, которые приводят к наблюдаемому актером результату.

На диаграмме вариантов использования указан один актер – «Сотрудник», который совершает все действия, выделенные при проектировании. Диаграмма вариантов использования представлена на рисунке.

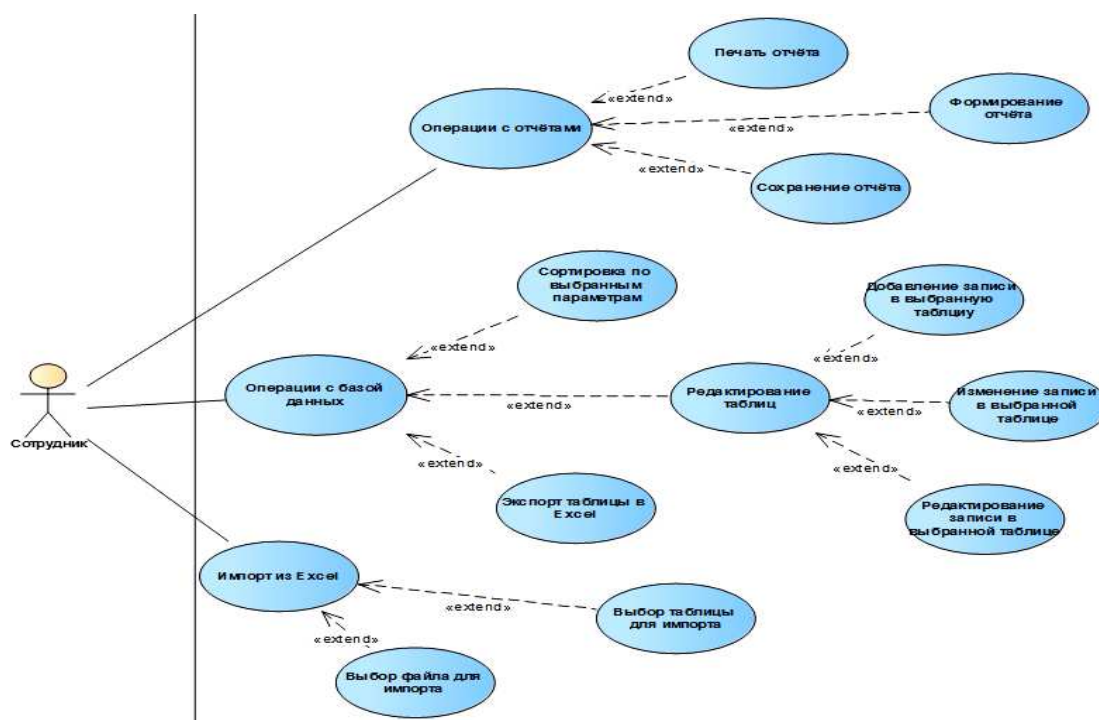


Диаграмма вариантов использования

Последний этап проектирования функциональной части программного обеспечения – *выбор инструментов*.

Выбор средств решения поставленной задачи исходил из того, какое программное и аппаратное обеспечение используется в данный момент на предприятии.

Выбор СУБД – сложная задача, должен основываться, в первую очередь, на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид программного продукта и категория пользователей. Другими показателями, влияющими на выбор СУБД, служат [2]:

- удобство и простота использования;
- качество средств разработки, защиты и контроля базы данных;
- уровень коммуникационных средств в случае применения ее в сетях;
- фирма-разработчик;
- стоимость.

Система MS SQL Server 2014 позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio. MS SQL Server обеспечивает высочайший уровень безопасности, надежности и масштабируемости для критически важных приложений. Чтобы использовать новые возможности, постоянно возникающие в быстро меняющемся деловом мире, предприятиям нужно быть способными быстро создавать и развертывать решения, управляемые данными. MS SQL Server 2014 позволяет сократить затраты времени и средств, требуемые на управление и развертывание таких приложений. Также следует учесть, что фирма-разработчик данной СУБД является также разработчиком самой распространенной ОС. В финансовом плане важным фактором является то, что существуют бесплатные сборки данной СУБД (Express).

Для реализации приложения была выбрана среда разработки Microsoft Visual Studio 2013, в качестве языка программирования – C#.

Достоинства платформы .NET [3]:

1 Вся платформа .NET основана на единой объектно-ориентированной модели. Все сервисы, интерфейсы и объекты, которые платформа предоставляет разработчику, объединены в единую иерархию классов. Другими словами, все, что может потребоваться при создании приложений под платформу .NET будет всегда у вас под рукой. Причем все это сгруппировано удобно и интуитивно понятно.

2 Приложение, написанное на любом .NET-совместимом языке, является межплатформенным (в идеале). Почему в идеале? Дело в том, что приложение, написанное, скажем, на том же C#, не зависит от платформы, на которой будет выполняться, но зато зависит от наличия платформы .NET.

3 В состав платформы .NET входит «сборщик мусора», который освобождает ресурсы. Таким образом, приложения защищены от утечки памяти и от необходимости освобождать ресурсы. Это делает программирование более легким и более безопасным.

4 Приложения .NET используют безопасные типы, что повышает их надежность и совместимость.

5 NET приложения могут быть сертифицированы на безопасность. Это является особенностью промежуточного кода, в который преобразуются все .NET приложения.

6 Абсолютно все ошибки обрабатываются механизмом исключительных ситуаций. Это позволяет избежать разногласий, которые иногда возникают при программировании под Win32.

7 Повторное использование кода стало еще удобнее. Это связано с тем, что промежуточный язык MSIL не зависит от языка программирования. Например, вы можете написать программу на C#, а патч к ней писать уже, скажем, на J#.

Заключение

В результате проведенной работы спроектировано веб-приложение, позволяющее хранить, изменять, добавлять информацию в базу данных, осуществлять быстрый поиск и различную группировку информации с последующим формированием различных отчетов.

ЛИТЕРАТУРА

1. Лекция 3: Элементы графической нотации диаграммы вариантов использования [Электронный ресурс]. – Режим доступа: <http://www.intuit.ru/studies/courses/32/32/lecture/1004?page=2>. – Дата доступа: 27.09.2017.
2. Выбор СУБД [Электронный ресурс]. – Режим доступа <http://websideworld.com/main/data-storage/6/2>. – Дата доступа: 27.09.2017.
3. Нгуен, П. Достоинства и недостатки .Net / Павел Нгуен [Электронный ресурс]. – Режим доступа: http://blog.nguen.net/post33-good_bad_plus_minus_dot_net.html. – Дата доступа: 27.09.2017.

УДК 004.021

**КОМБИНИРОВАННЫЕ АЛГОРИТМЫ ШИФРОВАНИЯ,
ЦЕЛЕСООБРАЗНОСТЬ ИХ ИСПОЛЬЗОВАНИЯ ДЛЯ ЗАЩИТЫ ФАЙЛОВ
ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА****А.Г. АНДРЕЙЧИКОВ****(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)**

Рассматриваются возможности комбинированных алгоритмов шифрования. Показаны преимущества и недостатки комбинированных алгоритмов при их использовании для защиты файлов от несанкционированного доступа.

Комбинированный (гибридный) метод шифрования позволяет сочетать преимущества высокой секретности, предоставляемые асимметричными криптосистемами с открытым ключом, с преимуществами высокой скорости работы, присущими симметричным криптосистемам с секретным ключом. При таком подходе криптосистема с открытым ключом применяется для шифрования, передачи и последующего дешифровки только секретного ключа симметричной криптосистемы. А симметричная криптосистема применяется для шифрования и передачи исходного открытого текста. В результате криптосистема с открытым ключом не заменяет симметричную криптосистему с секретным ключом, а лишь дополняет ее, позволяя повысить в целом защищенность передаваемой информации.

Для эффективного использования гибридных криптосистем необходимо рассмотреть отдельно симметричные и асимметричные алгоритмы шифрования их преимущества и недостатки.

Симметричные алгоритмы – это алгоритмы шифрования, в которых для шифрования и дешифровки используется один и тот же ключ, что, в свою очередь, обеспечивает высокую скорость работы.

Алгоритмы шифрования данных широко применяются в компьютерной технике в системах сокрытия конфиденциальной и коммерческой информации от злонамеренного использования сторонними лицами. Главным принципом в них является условие, что передатчик и приемник заранее знают алгоритм шифрования, а также ключ к сообщению, без которых информация представляет собой всего лишь набор символов, не имеющих смысла.

Классическими примерами таких алгоритмов являются симметричные криптографические алгоритмы:

- Простая перестановка.
- Одиночная перестановка по ключу.
- Двойная перестановка.
- Перестановка «Магический квадрат».

Основными достоинствами данного типа алгоритмов является скорость и простота реализации. Также стоит отметить, что при одинаковой длине ключа криптостойкость симметричных алгоритмов в несколько раз выше, чем асимметричных. Данный вид алгоритмов шифрования появился достаточно давно, и хорошо изучен.

Так как для шифрования и дешифровки используется один и тот же ключ, то самой большой проблемой при использовании данного типа алгоритмов является передача данного ключа. Передача ключа должна осуществляться по секретным каналам обеим сторонам. Эта проблема порождает ещё один важный недостаток – сложность управления ключами в случае большой сети, что сильно усложняет обмен данными и контроль за секретностью ключей.

Асимметричные криптографические системы были разработаны в 1970-х годах. Принципиальное отличие асимметричной криптосистемы от криптосистемы симметричного шифрования состоит в том, что для шифрования информации и ее последующего расшифровывания используются различные ключи:

- открытый ключ K используется для шифрования информации, вычисляется из секретного ключа k ;
- секретный ключ k используется для расшифровывания информации, зашифрованной с помощью парного ему открытого ключа K .

Эти ключи различаются таким образом, что с помощью вычислений нельзя вывести секретный ключ k из открытого ключа K . Поэтому открытый ключ K может свободно передаваться по каналам связи.

Асимметричные системы называют также двухключевыми криптографическими системами, или криптосистемами с открытым ключом.

По сравнению с симметричными алгоритмами асимметричные имеют ряд преимуществ, однако есть и недостатки. Главное преимущество состоит в том, что нет необходимости предварительно передавать секретный ключ. Также преимуществом асимметричных алгоритмов перед симметричными является тот факт, что при использовании системы в большой сети количество ключей значительно меньше, что, в свою очередь, упрощает их распространение. В качестве главного недостатка выделим сложность

внесения изменений в алгоритм, поскольку при использовании алгоритма в практических целях часто возникает необходимость изменения алгоритма под конкретную задачу или проблему. Также нужно обратить внимание на большую размерность ключей по сравнению с симметричными алгоритмами при одинаковой криптостойкости.

В качестве примера работы симметричного алгоритма реализована простая программа для шифрования введенной строки.

Листинг 1 – Пример алгоритма шифрования 3DES

```
public static string Encrypt(string source, string key)
{
    TripleDESCryptoServiceProvider desCryptoProvider = new TripleDESCryptoServiceProvider();
    MD5CryptoServiceProvider hashMD5Provider = new MD5CryptoServiceProvider();

    byte[] byteHash;
    byte[] byteBuff;

    byteHash = hashMD5Provider.ComputeHash(Encoding.UTF8.GetBytes(key));
    desCryptoProvider.Key = byteHash;
    desCryptoProvider.Mode = CipherMode.ECB; //CBC, CFB
    byteBuff = Encoding.UTF8.GetBytes(source);

    string encoded =
    Convert.ToBase64String(desCryptoProvider.CreateEncryptor().TransformFinalBlock(byteBuff, 0, byteBuff.Length));
    return encoded;
}

public static string Decrypt(string encodedText, string key)
{
    TripleDESCryptoServiceProvider desCryptoProvider = new TripleDESCryptoServiceProvider();
    MD5CryptoServiceProvider hashMD5Provider = new MD5CryptoServiceProvider();

    byte[] byteHash;
    byte[] byteBuff;

    byteHash = hashMD5Provider.ComputeHash(Encoding.UTF8.GetBytes(key));
    desCryptoProvider.Key = byteHash;
    desCryptoProvider.Mode = CipherMode.ECB; //CBC, CFB
    byteBuff = Convert.FromBase64String(encodedText);

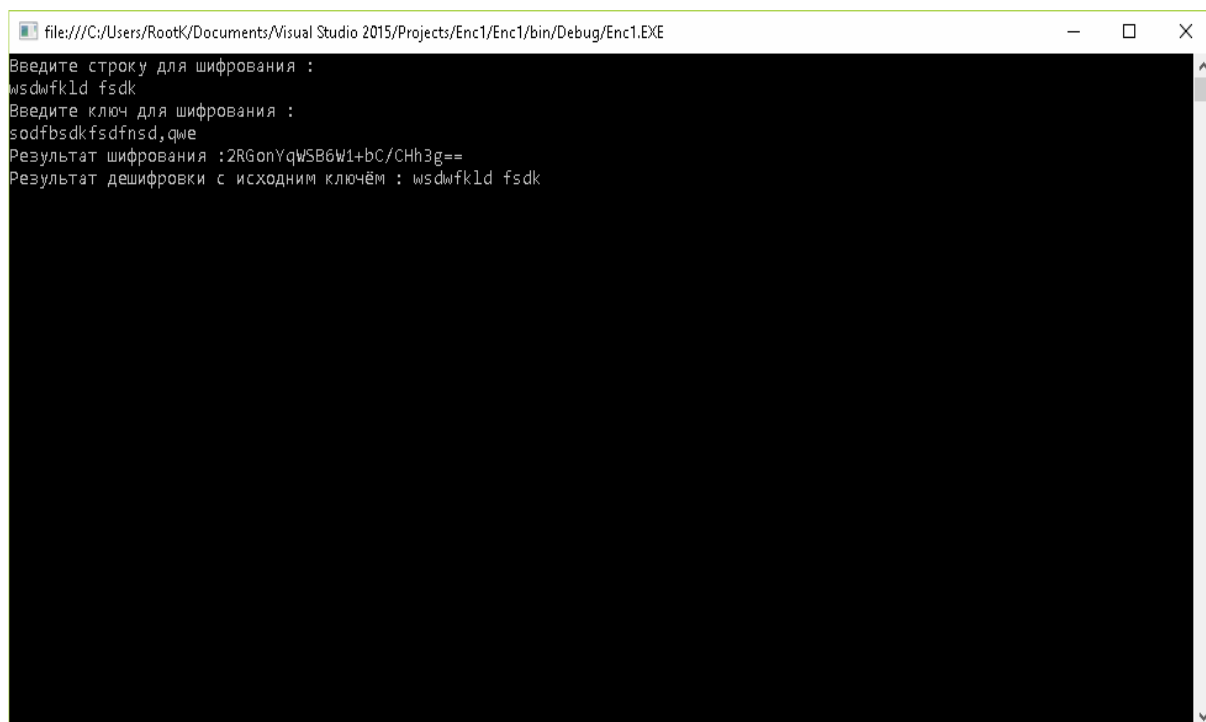
    string plaintext = Encoding.UTF8.GetString(desCryptoProvider.CreateDecryptor().TransformFinalBlock(byteBuff, 0, byteBuff.Length));
    byte[] byt = desCryptoProvider.CreateDecryptor().TransformFinalBlock(byteBuff, 0, byteBuff.Length);

    return plaintext;
}

static void Main(string[] args)
{
    Console.WriteLine("Введите строку для шифрования : ");
    String str = Console.ReadLine();
    Console.WriteLine("Введите ключ для шифрования : ");
    String key = Console.ReadLine();

    String result = Encrypt(str, key);
    Console.WriteLine("Результат шифрования : " + result);
    String deresult = Decrypt(result, key);
    Console.WriteLine("Результат дешифровки с исходным ключом : " + deresult);
    Console.ReadKey();
}
```

Результат работы алгоритма 3DES представлены на рисунке 1.



```
file:///C:/Users/RootK/Documents/Visual Studio 2015/Projects/Enc1/Enc1/bin/Debug/Enc1.EXE
Введите строку для шифрования :
wsdwfkld fsdk
Введите ключ для шифрования :
sodefbsdk fsdfnsd,qwe
Результат шифрования : 2RGonYqWSB6W1+bC/CHh3g==
Результат дешифровки с исходным ключём : wsdwfkld fsdk
```

Рисунок 1. – Результат работы алгоритма 3DES

Заключение

В данной статье представлен краткий обзор особенностей симметричных и асимметричных алгоритмов шифрования. Показаны преимущества и недостатки этих типов алгоритмов шифрования. В качестве примера симметричного типа шифрования представлен алгоритм 3DES и результаты работы простейшей программы с его использованием.

ЛИТЕРАТУРА

1. Хабрахабр [Электронный ресурс] Пара слов о гибридном шифровании и эллиптических кривых. – Режим доступа: <https://habrahabr.ru/post/106057>. – Дата доступа: 24.09.2017.
2. Хабрахабр [Электронный ресурс] Немного практической криптографии под .NET для чайников. – Режим доступа: <https://habrahabr.ru/post/254909>. – Дата доступа: 24.09.2017.
3. Хабрахабр [Электронный ресурс] RSA, а так ли все просто. – Режим доступа: <https://habrahabr.ru/post/99376>. – Дата доступа: 24.09.2017.

УДК 004.021

ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА ДЛЯ ЗАЩИТЫ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА К ФАЙЛАМ

А.Г. АНДРЕЙЧИКОВ*(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)*

Представлен практический способ создания приложения с использованием криптосистемы, основанной на гибридных алгоритмах шифрования. Результатом работы явилось написание программы для защиты пользовательских файлов с использованием гибридных алгоритмов шифрования, а также создание понятного пользовательского интерфейса. Задача реализации интерфейса решалась с использованием WindowsForm и языка программирования C#.

В настоящий момент самым распространенным способом хранения информации является запись ее на цифровые носители. Часть сохраняемой информации может быть конфиденциальной, приватной, секретной и нуждаться в защите. У такой информации есть законные пользователи. Но всегда существует вероятность появления пользователей незаконных, стремящихся захватить секретную информацию с целью обращения ее себе во благо. Хакеры ежедневно крадут номера кредитных карт, банковские счета и даже личность человека. Опасаясь этого, законные пользователи принимают меры по защите информации.

На сегодняшний день известны несколько подходов к проблеме защиты информации, хранящейся в персональном компьютере. Один из них – шифрование данных.

Принцип разработанного алгоритма защиты информации и его функционал. В качестве симметричного алгоритма шифрования использован алгоритм Triple DES (3DES). Triple DES (3DES) – симметричный блочный шифр, созданный Уитфилдом Диффи, Мартином Хеллманом и Уолтом Тачманном в 1978 году на основе алгоритма DES с целью устранения главного недостатка последнего – малой длины ключа (56 бит), который может быть взломан методом полного перебора ключа. Скорость работы 3DES в 3 раза ниже, чем у DES, но криптостойкость намного выше. 3DES является простым способом устранения недостатков DES. Алгоритм 3DES построен на основе DES, поэтому для его реализации можно использовать программы, созданные для DES.

В качестве асимметричного алгоритма шифрования выбран алгоритм RSA, который относится к так называемым асимметричным алгоритмам, у которых ключ шифрования не совпадает с ключом дешифровки. Один из ключей доступен всем и называется открытым ключом, другой хранится только у его хозяина и неизвестен никому другому. С помощью одного ключа можно производить операции только в одну сторону. Если сообщение зашифровано с помощью одного ключа, то расшифровать его можно только с помощью другого. Имея один из ключей, невозможно найти другой ключ, если разрядность ключа высока.

Для аутентификации пользователя используется система логин-пароль. В систему можно зайти с любой комбинацией логина-пароля, однако файлы, зашифрованные с помощью одной комбинации логина-пароля, невозможно дешифровать с помощью другой комбинации, даже введя верный ключ. Данный подход обеспечивает дополнительный уровень защиты от несанкционированного доступа к секретным данным.

Интерфейс программы создан с помощью WindowsForm на языке программирования C# (рис. 1).

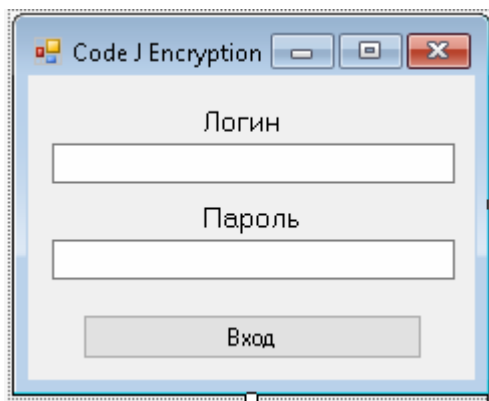


Рисунок 1. – Форма для авторизации в программе

После авторизации в программе и выбора действия интерфейс будет выглядеть, как показано на рисунках 2, 3.

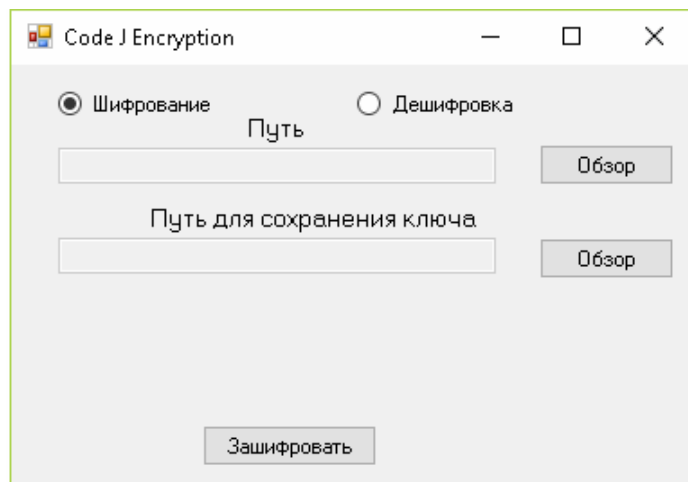


Рисунок 2. – Интерфейс шифрования

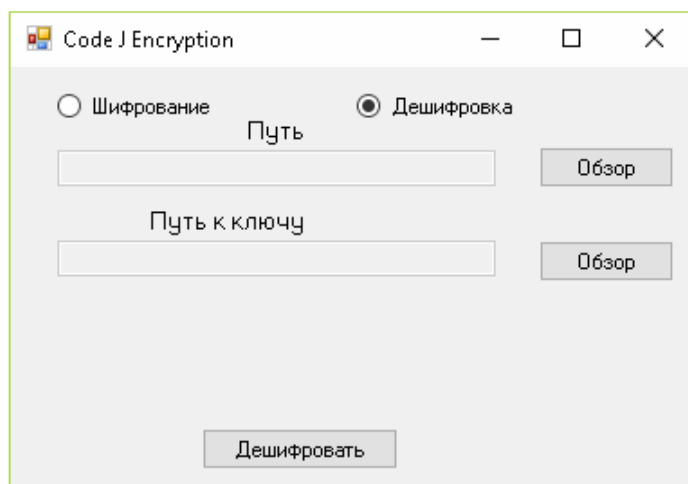


Рисунок 2. – Интерфейс дешифровки

Заключение

Автором разработан программный продукт для надежного хранения конфиденциальной, секретной информации на персональном компьютере. Ведутся работы по увеличению функционала программы и улучшению степени защиты данных.

ЛИТЕРАТУРА

1. Triple DES [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Triple_DES. – Дата доступа: 24.09.2017.
2. Хабрахабр [Электронный ресурс] Пара слов о гибридном шифровании и эллиптических кривых. – Режим доступа: <https://habrahabr.ru/post/106057>. – Дата доступа: 24.09.2017.
3. Хабрахабр [Электронный ресурс] Немного практической криптографии под .NET для чайников. – Режим доступа: <https://habrahabr.ru/post/254909>. – Дата доступа: 24.09.2017.
4. Хабрахабр [Электронный ресурс] RSA, а так ли все просто. – Режим доступа: <https://habrahabr.ru/post/99376>. – Дата доступа: 24.09.2017.

УДК 004.021

**ПРОЕКТИРОВАНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА
ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ «МУЗЫКАНТЫ РОССИИ»
ПОД ОПЕРАЦИОННУЮ СИСТЕМУ iOS**

А.В. ВЕЧЕРОВ*(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются принципы построения графического интерфейса пользователя в мобильном приложении под операционную систему iOS. Показаны их плюсы и минусы. Представлены примеры работы с каждой из систем.

В каждой операционной системе свои принципы построения графического интерфейса. В iOS для этого используются XML-файлы – Storyboards, либо программное создание разметки посредством кода. Расположение объектов формируется через задание рамки, координат и размера элементов интерфейса либо с помощью системы AutoLayout-ов, которые задают взаимные отступы.

Данная работа направлена на подробное описание технологий построения разметки в iOS.

Как правило, приложения в iOS строятся по шаблону MVC (Model-View-Controller или Модель-Вид-Контроллер). Идея данного шаблона проста – разделение обязанностей: задача контроллера – обработка действий пользователя (нажатия по кнопкам, обработка запросов к серверу и т.д.); модель предоставляет контроллеру представление данных, запрашиваемых пользователем; вид, в свою очередь, обеспечивает представление данных, полученных из модели.

Storyboard – файл разметки, который предоставляет разработчику удобный набор инструментов для построения графического интерфейса. Несмотря на то, что файл использует язык разметки XML, программисту не нужно его знать, так как принцип работы построен на перетаскивании элементов на рабочий стол файла и указании свойств данного объекта через меню.

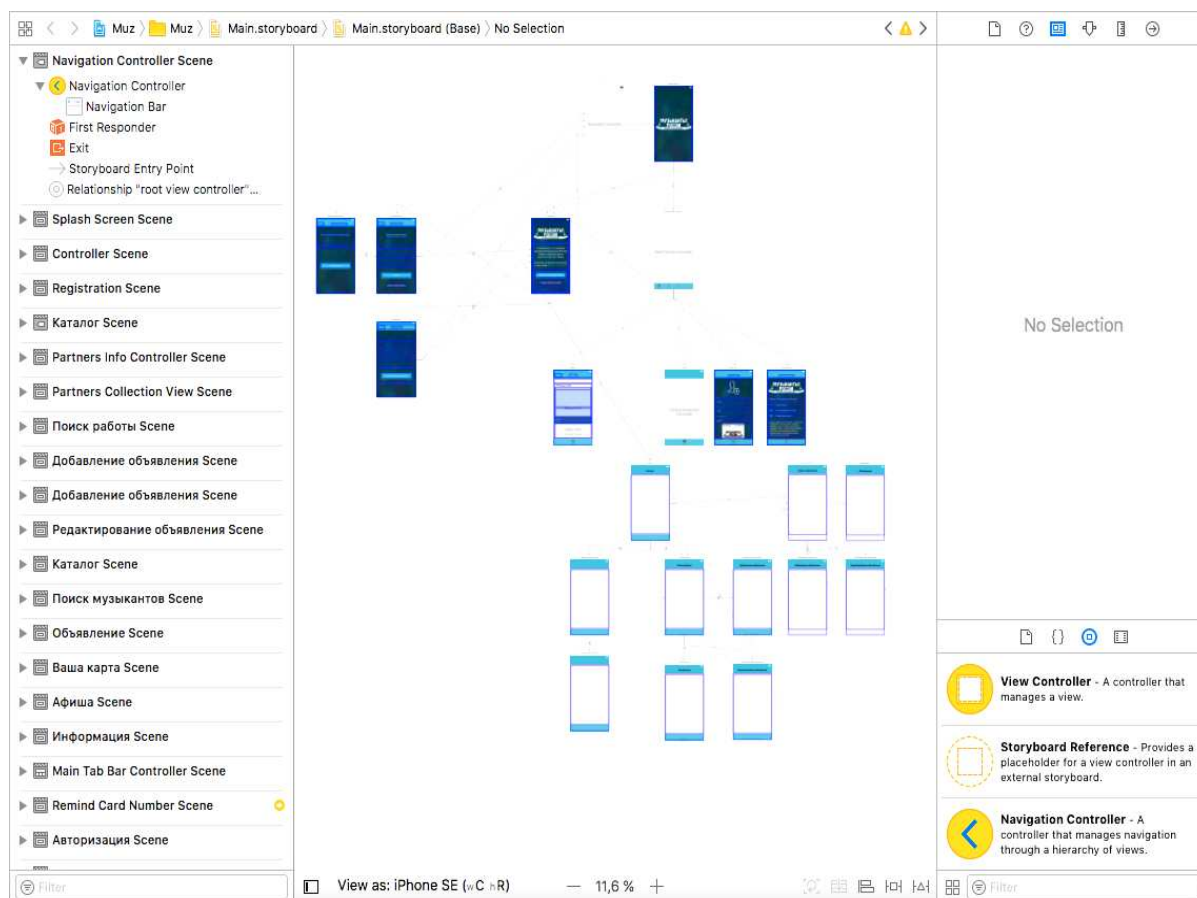


Рисунок 1. – Storyboard файл разметки

Данная технология имеет свои плюсы:

- наглядность и простота верстки;
- простое формирование переходов между экранами приложения;
- не нужно знать тип перехода, достаточно указать его идентификатор;
- SizeClasses – возможность задавать расположение элементов для различных экранов устройств

Однако есть и минусы:

- сложность при работе в команде, данные файлы плохо поддаются слиянию;
- большую часть кода анимации не вставить в данные файлы;
- большая нагрузка в случае увеличения размеров проекта;
- ограниченный выбор элементов разметки.

Как видно из рисунка 1, все переходы представлены стрелками между экранами, у каждого перехода есть свой идентификатор для доступа к ним. Также в правом нижнем углу расположено меню с выбором доступных элементов, однако список их ограничен.

Ручная разметка прописывается в файлах Controller-ов и View. Однако сделать это можно двумя способами:

- через фреймы;
- через систему Autolayout.

Работа с фреймами заключается в непосредственном указании координат расположения у элементов интерфейса, а также их размеров. Данный способ не является лучшим решением, так как он не предоставляет достаточной гибкости, т.е. при изменении ориентации экрана интерфейс не перестроится сам, придется предусмотреть и этот случай. Однако данный способ работает с ресурсоемкими элементами быстрее, чем Autolayout.

Autolayout – система работы с GUI, которая динамически вычисляет позицию и размер элементов, на основе определенных правил (constraints), заданных для этих элементов. Преимущество данной системы в гибкости, так как нет необходимости в подгонке размеров и положений элементов для различных экранов и состояний. Для облегчения работы с Autolayout существует множество открытых фреймворков: SnapKit, PureLayout и др.

Преимущество ручной разметки перед разметкой через Storyboard состоит в следующем:

- гибкость;
- быстрая работа;
- можно работать с расположением элементов в процессе работы с приложением;
- проще работать с анимацией;
- отсутствие ограничений в выборе элементов, так как можно написать свои элементы и использовать готовые.

Однако использование ручной разметки не дает наглядности, в приложении становится больше кода. Пример ручной разметки представлен в листинге 1.

Листинг 1 – пример ручной разметки

```
class VKCheckbox: UIView
{
    var line = VKCheckboxLine.Normal
    var button = UIButton()
    var checkmark = VKCheckmarkView()
    override init(frame: CGRect){
        super.init(frame: frame)
        self.setupView()
    }
    required init?(coder aDecoder: NSCoder){
        super.init(coder: aDecoder)
        self.setupView()
    }
    func setupView(){
        self.backgroundColor= UIColor.clear
        self.cornerRadius= 8
        self.borderWidth= 3
        self.borderColor= UIColor.darkGray
```

```
self.color = UIColor(red: 46/255, green: 119/255, blue: 217/255, alpha: 1)
self.checkmark.frame = self.bounds
self.checkmark.autoresizingMask = [.flexibleWidth, .flexibleHeight];
self.addSubview(self.checkmark)
self.button.frame = self.bounds
self.button.autoresizingMask = [.flexibleWidth, .flexibleHeight];
self.button.addTarget(self, action: #selector(VKCheckbox.buttonDidSelected), for: .touchUpInside)
self.addSubview(self.button)
}
override func layoutSubviews() {
    super.layoutSubviews()
    self.button.bounds = self.bounds
    self.checkmark.bounds = self.bounds
}
}
```

Заключение

Рассмотрены основные способы построения графического интерфейса для мобильных приложений под операционную систему iOS, а также их плюсы и минусы. Представлены примеры работы с каждой из систем.

ЛИТЕРАТУРА

1. Ruseller [Электронный ресурс] Частная коллекция качественных материалов для тех, кто делает сайты. – Режим доступа: <https://ruseller.com/lessons.php?id=666>. – Дата доступа: 26.09.2017.
2. Habrahabr [Электронный ресурс] Ручная верстка vsStoryboard/Nib. – Режим доступа: <https://habrahabr.ru/post/osbd/271355/>. – Дата доступа: 26.09.2017.
3. Apple Inc [Электронный ресурс] Understanding Auto Layout. – Режим доступа: <https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/AutoLayoutPG/index.html>. Дата доступа: 26.09.2017.
4. EnvatoTutsPlus [Электронный ресурс] iOS Fundamentals: Frames, Bounds and CGGeometry. – Режим доступа: <https://code.tutsplus.com/tutorials/ios-fundamentals-frames-bounds-and-cggeometry--cms-21196>. – Дата доступа: 26.09.2017.

УДК 004.021

**ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ
ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ «МУЗЫКАНТЫ РОССИИ»
ПОД ОПЕРАЦИОННУЮ СИСТЕМУ iOS**

А.В. ВЕЧЕРОВ*(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются проектирование базы данных для мобильного приложения «Музыканты России» под операционную систему iOS. Краткий анализ предметной области выявил необходимый набор сущностей. Делается вывод, что для разработки представленной базы данных удобно использовать СУБД MySQL, так как она является открытой и бесплатной, а также поддерживает множество типов данных необходимых при разработке.

Трудно представить хорошее клиент-серверное приложение без разработанной базы данных. Они позволяют удобно хранить большие объемы информации не на устройстве пользователя, а на удаленном сервере, который может эти данные предоставить по запросу от приложения.

Данная статья направлена на описание базы данных для разработанного приложения и методов ее проектирования.

Цель моделирования данных состоит в обеспечении разработчика концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Наиболее распространенным средством моделирования данных являются диаграммы «сущность-связь» (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.

Нотация ERD была впервые введена П. Ченом (Chen) и получила дальнейшее развитие в работах Баркера. Методология IDEF1, разработанная Т. Рэмей, основана на подходе П. Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме.

В настоящее время на основе совершенствования методологии IDEF1 создана ее новая версия – методология IDEF1X [1].

В результате краткого анализа предметной области выявлен необходимый набор сущностей:

- Partners (Партнеры);
- Partner_Cards (Карты пользователей);
- Search_Ad (Объявления о поиске музыкантов или группы);
- Work_Ad (Объявления о поиске работы);
- Click_Statistics (Статистика переходов пользователя по страницам информации о партнерах);
- Partner_Cities (Список городов);
- Partner_Categories (Список категорий);
- Partner_Likes (Рейтинг партнеров).

Связь – это ассоциирование двух или более сущностей. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

В любой связи выделяются два конца (в соответствии с существующей парой связываемых сущностей), на каждом из которых указывается:

- имя конца связи;
- степень конца связи (сколько экземпляров данной сущности связывается);
- обязательность связи (т.е. любой ли экземпляр данной сущности должен участвовать в данной связи) [2].

Для реализации приложения необходимо установить все связи между сущностями: необходимо рассмотреть всю информационную систему в совокупности и определить взаимное влияние сущностей.

Реляционная модель представляет базу данных в виде множества взаимосвязанных отношений (таблиц), которые используются для хранения информации об объектах, представленных в базе данных.

Концептуальная схема базы данных представлена на рисунке 1.

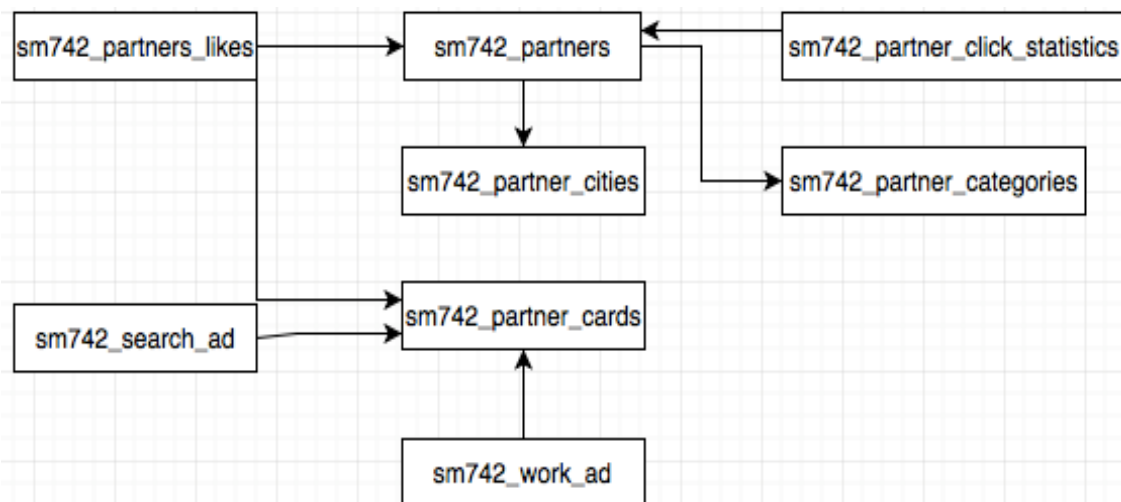


Рисунок 1. – Концептуальная схема базы данных

В соответствии выделенными выше сущностями определен набор необходимых таблиц базы данных, представленный в таблице 1.

Таблица 1. – Перечень таблиц базы данных

№ п/п	Название таблицы	Описание
1	Partners	Список всех партнеров
2	Partner_Cities	Список всех городов
3	Partner_Categories	Список всех категорий партнеров
4	Partner_Click_Statistics	Статистика переходов пользователя по страницам партнеров
5	Partners_Likes	Рейтинг партнеров
6	Partner_Cards	Список пользователей
7	Search_Ad	Список объявлений о поиске музыкантов или группы
8	Work_Ad	Список объявлений о работе

Важное достоинство баз данных – возможность хранить сгруппированные данные в разных таблицах и устанавливать связи между этими таблицами посредством ключей с последующим объединением этих данных в единую базу. Это позволяет снизить избыточность данных, которые хранятся, упростить организацию запросов для их получения.

Первичный ключ служит ограничителем целостности в рамках одной таблицы для идентификации, так как поле первичного ключа не может повторяться или быть пустым. Важное условие – каждая таблица может содержать только один первичный ключ.

Для организации связей между двумя и более таблицами используются вторичные ключи, которые служат ограничителями целостности связей нескольких таблиц, так как подчиненная таблица не может ссылаться на несуществующие записи главной таблицы (что позволяет строить целостные модели данных).

Существует три вида связей между таблицами:

- «один-ко-многим»;
- «один-к-одному»;
- «многие-ко-многим».

Связь «один-ко-многим» – самая распространенная для реляционных баз данных.

Для обеспечения ссылочной целостности в дочерней таблице создается внешний ключ, через который организовывается связь с дочерней таблицей.

Обязательное условие состоит в том, что внешний ключ по составу полей должен совпадать с первичным ключом родительской таблицы.

Первичные и внешние ключи отношений проектируемой базы данных представлены в таблице 2.

Таблица 2. – Первичные и внешние ключи отношений

№ п/п	Название таблицы	Первичный ключ	Внешние ключи
1	Partners	id	category city
2	Partner_Cities	id	
3	Partner_Categories	id	
4	Partner_Click_Statistics	id	partner_id
5	Partner_Cards	id	
6	Partners_Likes	id	partner_id user_id
7	Search_Ad	id	user_id
8	Work_Ad	id	user_id

Также важно выбрать систему управления базой данных.

Для разработки представленной базы данных удобно использовать СУБД MySQL, так как она является открытой и бесплатной, а также поддерживает множество типов данных, необходимых при разработке.

Заключение

Рассмотрены основные цели и принципы моделирования базы данных для приложения «Музыканты России».

Проанализированы основные сущности спроектированной базы данных.

ЛИТЕРАТУРА

1. Help4Students [Электронный ресурс] Методология моделирования данных ИС. – Режим доступа: [https://hlp4asp.wordpress.com/2011/10/13/методология-моделирования-данных-ис.](https://hlp4asp.wordpress.com/2011/10/13/методология-моделирования-данных-ис/) – Дата доступа: 27.09.2017.
2. CitForum [Электронный ресурс] Семантическое моделирование данных, ER-диаграммы. – Режим доступа: http://citforum.ru/database/osbd/glava_24.shtml. – Дата доступа: 27.09.2017.
3. Dev DocumentationMySQL[Электронный ресурс] MySQLWorkbench. – Режим доступа: <https://dev.mysql.com/doc/workbench/en/>. – Дата доступа: 27.09.2017.

УДК 004.021

**РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ ИНТЕРНЕТ-МАГАЗИНОМ,
ВЗАИМОДЕЙСТВИЕ СИСТЕМЫ С ПОЛЬЗОВАТЕЛЕМ****А.В. ГУБКО****(Представлено: Д.В. ПЯТКИН)**

Представлен практический способ создания простой и интуитивно понятной системы управления для интернет-магазина. Цель работы состояла в написании приложения для упрощения взаимодействия пользователя с базой данных при работе с наполнением интернет-магазина, для упрощения пользования самим интернет-магазином, а также в создании простейшего и понятного интерфейса управления и разработка защиты от несанкционированного доступа со стороны. Задача решалась посредством скриптового языка сценариев PHP.

В настоящее время человек старается оцифровать как можно больший объем информации, что позволит иметь к ней доступ с любого цифрового устройства в любой точке мира, имея лишь доступ к сетевым ресурсам. Это позволяет реализовать самые обычные и привычные всем действия более быстро. Например, совершать покупки. Но мало кто задумывался, что за системой, занимающейся продажами стоит сложная логика обработки заказов и демонстрации товаров. Именно для управления заказами и наполнением магазина занимается система управления. Это значит, что данная система оперирует большими объемами данных, включая платежные данные, личные данные покупателей, а также зачастую данные самого магазина и данные о товарах. Из этого следует, что рассматриваемая система должна отвечать некоторым требованиям к безопасности и предоставлять доступ узкому кругу лиц.

С учетом растущих требований у населения возрастают также требования к интерфейсу разрабатываемых продуктов. Как и в любом веб-приложении, интерфейс позволяет пользователю получать требуемую информацию в приемлемом виде.

Выбор технологий

Данная часть приложения является неотъемлемой частью интернет-магазина, следовательно, она также является веб-приложением. Именно поэтому разработка велась на языке PHP. Данный язык позволил реализовать работу с доступом к информации в базе данных. Данный язык предоставляет возможность быстро задать подключение к базе данных.

Помимо языка PHP, для улучшения внешнего вида интерфейса использовались такие языки, как Javascript и CSS.

CSS – формальный язык описания внешнего вида документа, написанного с помощью языка разметки. Служит в основном для создания каскадных таблиц стилей.

Javascript – язык программирования, придающий страницам интерактивность; служит для создания приложений и частей приложений.

Проектирование интерфейса

При разработке интерфейса, для удобства использования был применен способ группировки задач по блокам с определенной тематикой. Данный вид группировки позволяет упростить и ускорить поиск блоков, а также систематизировать данные на экране, для лучшей читабельности и понимания.

Сравнение доступа через систему и через базу

Для добавления информации в интернет магазин напрямую через базу данных пользователю в определенных случаях необходимо добавлять связи и описание сразу в большое количество таблиц, как показано на рисунке 1.

Как видно из рисунка, для добавления информации необходимо задействовать большое количество таблиц. Также при оформлении заказа, когда информация требует обновления, придется искать необходимую информацию во всех таблицах. Это существенно осложняет работу с наполнением интернет-магазина и с самим процессом работы с ним. Именно для этих целей и разрабатывается система управления содержимым интернет-магазина (рисунок 2).

На рисунке 2 можно видеть, что тот же процесс добавления данных, но используя систему управления, становится гораздо проще и быстрее. Простота достигается тем, что все связи задаются заранее через программный код, и пользователю не надо отслеживать все таблицы на предмет наличия данных, система сделает все сама. Пользователю остается только ввести необходимые данные для добавления или выбрать данные, требующие изменения. Доступ напрямую через базу очень сложен, и даже если пользователь обладает всеми необходимыми данными о структуре базе данных, данный процесс все равно остается достаточно длительным. Также никто не застрахован от человеческих ошибок и обычной

невнимательности, что может повлечь за собой потерю данных, и в итоге – возможную потерю прибыли от интернет-магазина.

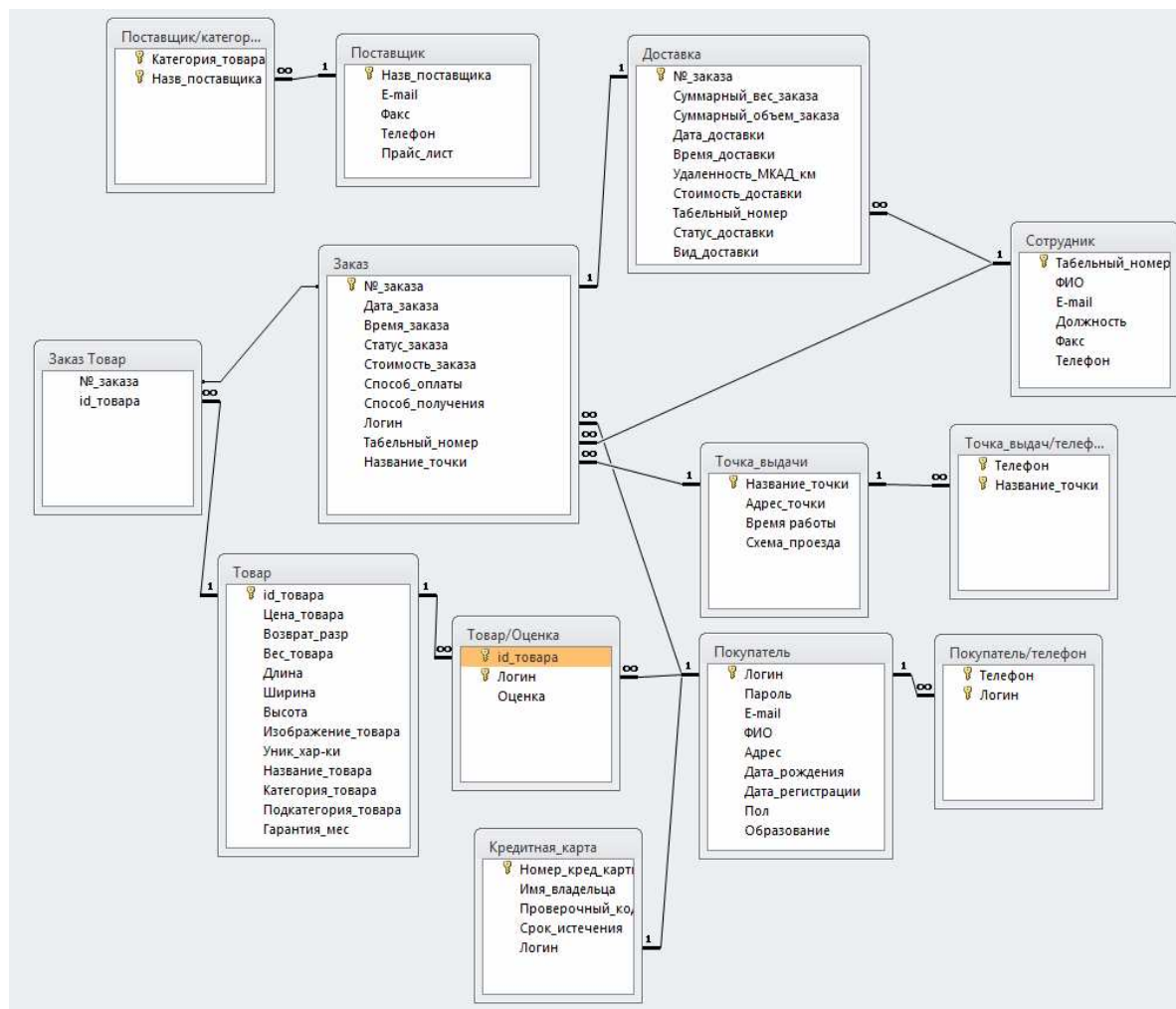


Рисунок 1. – Количество таблиц в базе данных интернет магазина

Новая сделка

☰

Связать с интернет-магазином: ☒

☰

О сделке

☰

Название:

Стадия сделки:

Валюта:

Рисунок 2. – Добавление информации в базу через интерфейс

Защита данных от несанкционированного доступа

Так как данная система оперирует всеми данными, которые находятся в базе данных (данные пользователей, счета, номера карт, пароли), то при их утрате могут возникнуть довольно большие проблемы.

Исходя из вышесказанного, можно сделать вывод, что данные подлежат защите. Самым простым методом защиты может стать закрытие прямого доступа к системе, чтобы никто не смог получить доступ к данным. Но тогда возникает ситуация, что доступ должен быть у определенного круга лиц, которые могут работать в системе. Решением данной проблемы является создание группы пользователей, которой выдан доступ для работы с данной системой. Это позволяет ограничить доступ для сторонних лиц и предоставить доступ к системе выделенному кругу лиц. Но иногда этого может быть недостаточно, и сторонние лица могут завладеть правами доступа к системе. В этом случае потребуется вторая ступень защиты, которая заключается в шифровании паролей и номеров карт, чтобы никто не имел возможности видеть «открытые» данные о паролях и номерах карт. Также сама система может являться ступенью защиты, так как с базой данных взаимодействует не сам пользователь, а сама система. Защита данных с помощью системы объясняется тем, что при взаимодействии с базой пользователь может ввести некорректные данные и тем самым повлечет как минимум наличие некорректных данных, а как максимум – очистку или даже удаление базы данных. Система же должна проводить проверку вводимых данных на корректность, запрашивать повторный ввод данных либо производить редактирование самостоятельно.

Взаимодействие с пользователем

Система должна взаимодействовать с пользователем для облегчения поиска информации и увеличения производительности. В случае если система позволяет оформлять заказы, пользователь должен видеть, что поступил заказ. Система должна уведомлять об этом. В данном случае пользователю не придется самому просматривать данные в поисках заказа, так как система должна автоматически сделать это сама. Это пример сложного взаимодействия с системой, однако есть и более простые примеры. В системе имеются блоки, описывающие данные из таблиц базы данных, и пользователь интуитивно понимает, с какой частью системы он работает в данный момент. Также во взаимодействии с пользователем существенную роль играет наличие интерфейса, который позволяет обеспечить удобный и понятный процесс работы в самой системе.

Заключение

Разработка системы управления является неотъемлемой частью создания интернет-магазина, так как позволяет не только упростить и ускорить работу в самой системе, но и защитить данные пользователя, клиентов и данные самого магазина. Разработка же интерфейса в существенной мере упрощает понимание элементов системы, что также облегчает дальнейшую работу.

ЛИТЕРАТУРА

1. PHP. Hypertext Preprocessor [Электронный ресурс]. – Режим доступа: <http://php.net/manual/ru/intro-whatis.php>. – Дата доступа: 28.09.2017.
2. Материал из Википедии – свободной энциклопедии CSS [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/CSS>. – Дата доступа: 28.09.2017.
3. Javascriptто это такое? | Все что вы хотели знать о хостинге и сайтостроении [Электронный ресурс]. – Режим доступа: <http://articles-hosting.ru/329/javascript-%E2%80%93-chto-eto-takoe.html>. – Дата доступа: 28.09.2017.

УДК 004.021

**ЭКОНОМИЧЕСКАЯ СОСТАВЛЯЮЩАЯ РАЗРАБОТКИ ИНТЕРНЕТ-МАГАЗИНА,
ОЦЕНКА ВЛИЯНИЯ ИНФОРМАЦИОННОЙ СРЕДЫ НА ЭКОНОМИКУ****А.В. ГУБКО***(Представлено: Д.В. ПЯТКИН)*

Представлена информация об экономической части разработки интернет магазина. Также представлена информация об основных стадиях разработки приложения и их экономической составляющей. Проводится сравнительный анализ интернет магазинов с обычными магазинами. Сравниваются их достоинства и недостатки. Оценивается экономическая эффективность.

В настоящее время стремительными темпами развивается сфера интернет-торговли. Все большее количество людей предпочитают совершать покупки онлайн, не выходя из дома, даже если учесть тот факт, что покупатель не может полностью оценить товар перед покупкой, существует риск быть обманутым. Однако все это не отпугивает людей от совершения покупок через интернет. При покупке товаров через интернет покупатели могут руководствоваться различными сервисами, предоставляющими возможность обмена, впечатлениями различных реальных пользователей по поводу того или иного интернет-магазина.

Стадии разработки:

С экономической точки зрения разработка любого приложения, будь то магазин или игра, делится на 4 важнейших этапа:

- 1) проектирование;
- 2) реализация;
- 3) тестирование;
- 4) доработка.

На этапе *проектирования* проводится анализ для дальнейшей разработки. Уже на этом этапе закладываются основные требования и определяются примерные объемы работ и затраты на них.

К этапу *реализации* относится непосредственно сама разработка продукта. Данный этап является самым долгим и объемным, также он включает в себя наибольшие затраты на саму разработку.

На этапе *тестирования* проводится тестирование разработанного продукта с целью выявления неточностей и погрешностей в логике или методах решения поставленных задач. Этап доработки включает финальные испытания и изменения системы, а также работы по приемке-сдаче продукта в оборот. Затраты на каждом из этапов проектирования можно увидеть на рисунке 1.

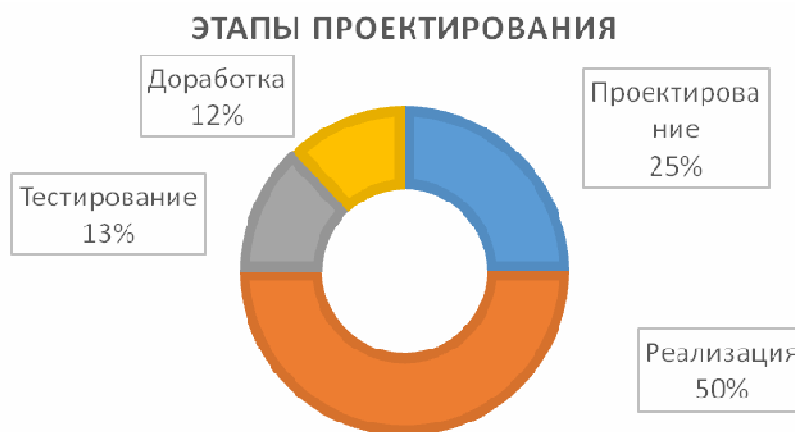


Рисунок 1. – Затраты на этапах проектирования

Расчет стоимости продукта

В стоимость любого продукта включена стоимость его создания. В данном случае, программным продуктом является приложение, интернет-магазин. Расчет стоимости должен учитывать следующие статьи затрат:

- 1) *материалы и комплектующие* – расходные материалы, потраченные при создании продукта;

- 2) *электроэнергия* – так как продуктом является приложение, то при его разработке используют компьютеры, потребляющие электричество;
 - 3) *основная заработная плата исполнителей* – затраты на выдачу заработной платы сотрудникам, работающим над проектом;
 - 4) *дополнительная заработная плата* – затраты на оплату премий, рисков, отпусков и прочее;
 - 5) *отчисления на социальные нужды* – затраты на налоговые сборы, в фонд защиты населения, страхование, прочее;
 - 6) *амортизация* – затраты на непредвиденные обстоятельства;
 - 7) *расходы на спецоборудование* – затраты, связанные с приобретением специализированного оборудования;
 - 8) *накладные расходы* – расходы на общехозяйственные нужды;
 - 9) *прочие прямые расходы* – затраты на подготовку и приобретение различного рода информации.
- Основные затраты на разрабатываемый продукт, вне зависимости от сроков разработки, приходятся на выдачу заработной платы сотрудникам.

График распределения затрат можно наблюдать на рисунке 2.

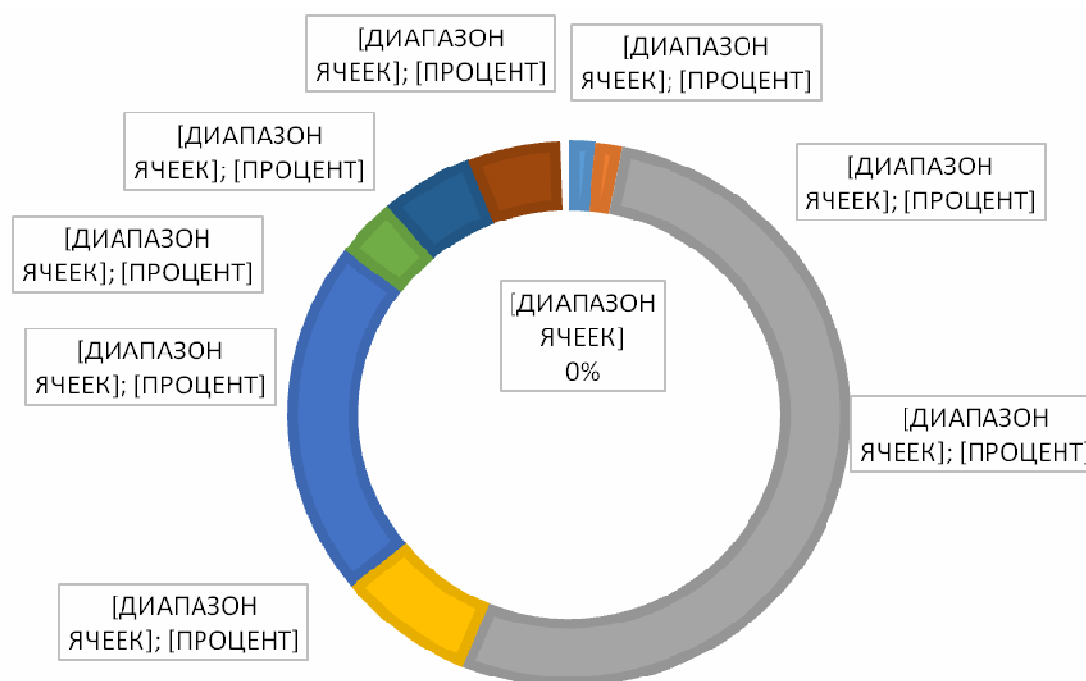


Рисунок 2. – структура затрат на разработку

В каждом отдельном случае данные могут меняться, однако при любых обстоятельствах основными затратами будет являться заработная плата сотрудникам, выполняющим работы по разработке продукта. Эти факторы следует учитывать при планировании каждого отдельного проекта, и при планировании времени, за которое планируется погасить расходы и выйти в прибыль.

Сравнительный анализ интернет магазинов с обычными магазинами

В данный момент появилась тенденция, согласно которой интернет-магазины становятся более популярными, чем обычные магазины. Это существенно увеличивает объемы продаж, а соответственно и прибыль интернет-магазинов.

На рисунке 3 можно видеть, насколько увеличились объемы продаж в интернет-магазинах за последние годы. В это же время популярность обычных магазинов начинает падать, и чтобы не потерять клиентов, обычные магазины начинают внедрять интернет-ресурсы в свою деятельность. Например, онлайн-заказ продуктов и онлайн-доставка. Однако не стоит забывать, что существенным преимуществом обычных магазинов является наглядное представление товара и возможность самостоятельного выбора.

Достоинства и недостатки:

Каждый из видов магазинов имеет свои плюсы и минусы.

Достоинства интернет-магазина:

+ малый стартовый капитал;

- + сравнительно недорогое содержание магазина;
- + проведение аналитики;
- + спрос, обусловленный растущим числом потребителей;
- + более низкая цена, за счет малых затрат на содержание;

Достоинства обычного магазина:

- + возможность личной оценки товара;
- + моментальность покупки;
- + возможность возврата;
- + большее доверие клиентов.

В данном случае достоинства одного вида магазина являются недостатком магазина другого вида.



Рисунок 3. – Объемы продаж интернет магазинов

Оценка экономической эффективности:

Как было упомянуто ранее, интернет-магазины приобретают все большую популярность, в то время как обычные магазины ее теряют. Этот факт существенно влияет на объемы продаж и объемы прибыли. Именно поэтому разработка и содержание интернет-магазина является более экономически выгодным, нежели открытие обычного магазина.

Заключение

Экономика развивается совместно с информационными технологиями. Разработка интернет-магазинов является экономически выгодным, т.к. в данный момент все больше клиентов хотят упростить жизнь, и сделать ее комфортной. С помощью интернет-магазинов можно экономит не только деньги, но и время. Интернет-магазины позволяют избавиться от утомительного ожидания в очередях. Однако помимо всего, у интернет-магазинов имеется и ряд недостатков, основным из которых является отсутствие возможности полного просмотра приобретаемого товара.

ЛИТЕРАТУРА

1. Insales.ru [Электронныйресурс]. – Режим доступа: <https://www.insales.ru>. – Дата доступа: 28.09.2017.
2. AvenueSoft.ru [Электронныйресурс]. – Режим доступа: http://avenuesoft.ru/shop/internet_magazin_vs_offline_magazin.html. – Дата доступа: 28.09.2017.
3. Richpro.ru – финансовый журнал [Электронныйресурс]. – Режим доступа:<http://richpro.ru>. – Дата доступа: 28.09.2017.

УДК 004.932.72

**ОБНАРУЖЕНИЕ ЛИЦ НА ИЗОБРАЖЕНИЯХ
МЕТОДОМ ГИСТОГРАММЫ НАПРАВЛЕННЫХ ГРАДИЕНТОВ****И.И. ИВАНОВ***(Представлено: Е.С. ГАТИХО)*

Рассматривается возможность обнаружения и локализации лиц на изображениях с помощью применения метода, основанного на построении гистограммы направленных градиентов в совокупности с методом скользящего окна.

В задачах, связанных с машинным зрением, одной из самых ресурсоемких подзадач остается обнаружение и локализация объекта на изображении для последующей обработки выделенного участка алгоритмами классификации. Метод, основанный на построении гистограммы направленных градиентов, является одним из самых эффективных решений, используемых для обнаружения объектов.

Основной идеей данного алгоритма является допущение, что внешний вид и форма объекта на фрагменте изображения могут быть описаны распределением градиентов интенсивности. При описании фрагмента изображения оно разбивается на несколько небольших участков, называемых ячейками. В ячейках вычисляются гистограммы направленных градиентов внутренних пикселей. Таким образом, конечный дескриптор, являющийся комбинацией всех гистограмм, отображает направления перехода от света к темноте по всему изображению, показывая упрощенное представление объектов на нем, независимо от его яркости [1]. Пример гистограммы направленных градиентов представлен на рисунке 1.



Рисунок 1. – Пример гистограммы направленных градиентов

Как видно, объекты на гистограмме упрощенно представляются множеством направлений градиентов, что позволяет выделить их формы и, соответственно, примерный внешний вид искомого объекта. Этого вполне достаточно для его легкой локализации с использованием канонического шаблона, составленного с использованием некоторой выборки изображений, непосредственно содержащих искомый объект в необходимой проекции.

При вычислении градиентов производится свертка изображения с ядрами $[-1, 0, 1]$ и $[-1, 0, 1]^T$, в результате чего образуются две матрицы D_x и D_y производных вдоль осей x и y соответственно. Эти матрицы используются для вычисления углов и модулей градиентов в каждой точке изображения.

Каждому интервалу ставится в соответствие бин гистограммы. Тогда гистограмма ячейки заполняется так, что величина градиента в каждой внутренней его точке добавляется к величине бина, соответствующего интервалу, содержащему угол данного градиента [2].

Дескриптор гистограммы ориентированных градиентов имеет несколько преимуществ над другими дескрипторами. Поскольку он работает локально, метод поддерживает инвариантность геометриче-

ских и фотометрических преобразований, за исключением ориентации объекта. Подобные изменения появятся только в больших фрагментах изображения.

Для непосредственного обнаружения и локализации объекта разумно воспользоваться методом скользящего окна, заключающегося в отыскании такого участка изображения, который наиболее похож на известный шаблон объекта. Пример шаблона человеческого лица, составленного с помощью набора изображений лиц в фронтальной проекции, представлен на рисунке 2.



Рисунок 2. – Пример шаблона человеческого лица

Метод скользящего окна использует проход изображения областью, последовательно меняющей свое положение и размер, а также нахождение некоторого коэффициента схожести участка изображения, над которым на данном шаге находится текущее «окно» и шаблона искомого объекта.

Также иногда применяются некоторые модификации и улучшения метода скользящего окна в виде, например, динамического шага перемещения или коэффициента масштабирования окна в зависимости от содержимого изображения в соседних областях, что позволяет исключить излишние вычислительные операции и ускорить процесс обнаружения [2].

Также операции, связанные с процессом обнаружения, могут быть разбиты на несколько независимых друг от друга частей и произведены в несколько вычислительных потоков, что только ускорит работу алгоритма.

Следует отметить, что обнаружение таких объектов, как человеческие лица, не требует наличия множества деталей на изображении, что, безусловно, дает возможность масштабировать крупные изображения, многократно выигрывая в производительности, не теряя процента обнаружения, если не требуется локализация лиц относительно малых размеров.

Исходя из вышеописанного можно сделать следующие **выводы**. Применение метода обнаружения объектов на изображении, основанного на построении гистограммы направленных градиентов для обнаружения и локализации лиц, позволяет решать поставленную задачу с высокой точностью, не прибегая к высоким затратам вычислительных ресурсов. Это дает возможность построения систем обнаружения лиц не только на статичных изображениях, но и на множестве кадров, образующих видеопоток. Более того, с применением вышеописанных оптимизаций, модификаций и ухищрений, нацеленных на ускорение работы метода скользящего окна, метод, использующий построение гистограммы направленных градиентов, позволяет создавать легкие и эффективные системы обнаружения лиц даже для мобильных платформ, ограниченных в вычислительных мощностях, без ощутимого снижения эффективности.

Вкупе с невысокой чувствительностью к силе освещения, метод, основанный на построении гистограммы направленных градиентов, является одним из самых эффективных и популярных в сфере обнаружения и локализации объектов, задаваемых единым каноничным шаблоном, сгенерированным на небольшом множестве примеров изображений.

ЛИТЕРАТУРА

1. Гистограмма направленных градиентов [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Гистограмма_направленных_градиентов. – Дата обращения: 01.09.2017.
2. Южаков, Г.Б. Алгоритм быстрого построения дескрипторов изображения, основанных на технике гистограмм ориентированных градиентов / Г.Б. Южаков // ТРУДЫ МФТИ. – 2013. – Т. 5, № 3. – С. 87.

УДК 004.021

КЛАССИФИКАЦИЯ ДАННЫХ МЕТОДОМ ОПОРНЫХ ВЕКТОРОВ

И.И. ИВАНОВ

(Представлено: Е.С. ГАТИХО)

Анализируется классификация данных методом опорных векторов: описываются причины использования, описание алгоритма, его плюсы и минусы. Применение метода опорных векторов позволяет создать высокоточные системы классификации данных, имеющих место в статистическом анализе и методах машинного зрения.

Классификация данных – одна из центральных задач машинного обучения. В данном направлении интенсивно применяются методы оптимизации и аналитической геометрии. Задача классификации данных состоит в нахождении ответа на вопрос, к какому из определенных классов будут принадлежать новые наблюдаемые данные, основываясь на уже имеющемся наборе данных, классы которых известны. Каждый объект данных представляется как вектор (точка) в n -мерном пространстве (упорядоченный набор n чисел). Каждая из этих точек принадлежит только одному из определенных классов. Вопрос состоит в том, можно ли разделить точки гиперплоскостью размерности $n - 1$. Это типичный случай линейной разделимости. Искомых гиперплоскостей может быть множество, поэтому полагают, что максимизация зазора между классами способствует более уверенной классификации. То есть, можно ли найти такую гиперплоскость, чтобы расстояние от нее до ближайшей точки было максимальным. Это эквивалентно тому, что расстояние между двумя ближайшими точками, лежащими по разные стороны гиперплоскости, максимально. Если такая гиперплоскость существует, она называется оптимальной разделяющей гиперплоскостью, а соответствующий ей линейный классификатор называется оптимально разделяющим классификатором [1].

Пример иллюстрации оптимальной и неоптимальных разделяющих плоскостей в рамках абстрактного набора данных, заданного множеством векторов размерности 2, приведен на рисунке 1.

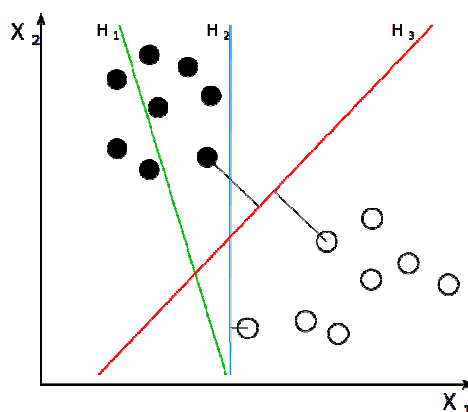


Рисунок 1. – Пример оптимальных и неоптимальных разделяющих плоскостей

При описании задачи полагается, что точки имеют вид:

$$\{(c_1 x_1), (c_2 x_2), \dots, (c_p x_p)\},$$

где c_i принимает значение 1 или -1 , в зависимости от того, какому классу принадлежит точка x_i .

Каждое x_i – это n -мерный вещественный вектор, обычно нормализованный значениями $[0,1]$ или $[-1,1]$. Если точки не будут нормализованы, то точка с большими отклонениями от средних значений координат точек слишком сильно повлияет на классификатор. Можно рассматривать это как учебную коллекцию, в которой для каждого элемента уже задан класс, к которому он принадлежит.

Для того чтобы метод опорных векторов классифицировал их таким же образом, строится разделяющая гиперплоскость, имеющая вид:

$$w \cdot x - b = 0.$$

Вектор w – перпендикулярен к разделяющей гиперплоскости.

Параметр $\frac{b}{\|w\|}$ равен по модулю расстоянию от гиперплоскости до начала координат.

Если параметр b равен нулю, гиперплоскость проходит через начало координат, что ограничивает решение. Так как производится поиск оптимального разделения, выделяются опорные векторы и гипер-

плоскости, параллельные оптимальной и ближайшие к опорным векторам двух классов. Можно показать, что эти параллельные гиперплоскости могут быть описаны следующими уравнениям (с точностью до нормировки):

$$w \cdot x - b = 1,$$

$$w \cdot x - b = -1.$$

Если обучающая выборка линейно разделима, то мы можем выбрать гиперплоскости таким образом, чтобы между ними не лежала ни одна точка обучающей выборки, и затем максимизировать расстояние между гиперплоскостями [2].

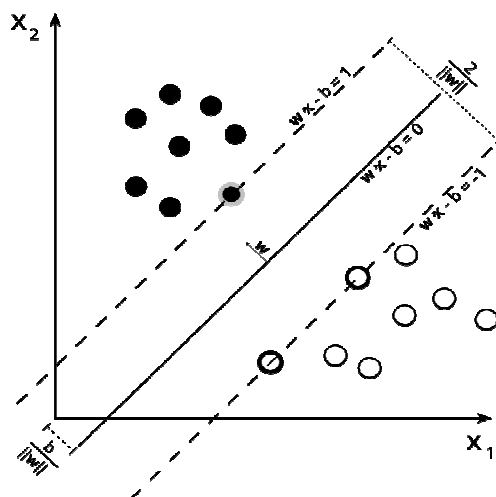


Рисунок 2. – Пример оптимальной гиперплоскости с выделенными опорными векторами

Ширину полосы между ними легко найти из соображений геометрии, она равна $\frac{2}{\|w\|}$. Таким образом, наша задача минимизировать $\|w\|$. Чтобы исключить все точки из полосы, мы должны убедиться для всех i , что и

$$w \cdot x_i - b \geq 1, \quad c_i = 1,$$

$$w \cdot x_i - b \leq -1, \quad c_i = 1.$$

Это может быть также записано в виде:

$$c_i (w \cdot x_i - b) \geq 1, 1 \leq i \leq n.$$

Метод опорных векторов обладает следующими преимуществами [2]:

- наиболее быстрый метод нахождения решающих функций;
- метод сводится к решению задачи квадратичного программирования в выпуклой области, которая всегда имеет единственное решение;
- метод находит разделяющую полосу максимальной ширины, что позволяет в дальнейшем осуществлять более уверенную классификацию;
- возможность классификации среди множества классов;
- малые временные затраты непосредственно на классификацию данных.

Однако, как и любой алгоритм, метод опорных векторов не лишен и своих недостатков, главным из которых является чувствительность к шумам и стандартизации данных.

Исходя из вышесказанного, можно сделать *вывод*, что применение метода опорных векторов позволяет создать высокоточные системы классификации данных, имеющих место в статистическом анализе и методах машинного зрения.

ЛИТЕРАТУРА

1. Support vector machine [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Support_vector_machine. – Дата доступа: 02.09.2017.
2. Машина опорных векторов [Электронный ресурс]. – Режим доступа: <http://www.machinelearning.ru/wiki/index.php?title=SVM>. Дата обращения: 02.09.2017.

УДК 004.056.55

СИММЕТРИЧНЫЕ И АСИММЕТРИЧНЫЕ ШИФРЫ

Р.Ю. КАРАБАНОВ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматриваются и сравниваются два самых распространенных типа шифров – симметричные и асимметричные. Выделены их достоинства и недостатки. Для криптографического закрытия и последующего расшифровывания передаваемой информации используются открытый и секретный ключи.

Шифр (от фр. chiffrer «цифра» от араб. صفر, *sifr* «ноль») – какая-либо система преобразования текста с секретом (ключом) для обеспечения секретности передаваемой информации. Шифр может представлять собой совокупность условных знаков (условная азбука из цифр или букв) либо алгоритм преобразования обычных цифр и букв.

По количеству используемых ключей шифры разделяются на *симметричные* (используют один ключ для шифрования и дешифрования) и *асимметричные* (используют два различных ключа) [1].

Симметричные шифры

Исторически первыми появились симметричные криптографические системы. В симметричной криптосистеме шифрования используется один и тот же ключ для зашифровывания и расшифровывания информации. Это означает, что любой, кто имеет доступ к ключу шифрования, может расшифровать сообщение.

Соответственно, с целью предотвращения несанкционированного раскрытия зашифрованной информации все ключи шифрования в симметричных криптосистемах должны держаться в секрете. Именно поэтому симметричные криптосистемы называют *криптосистемами с секретным ключом* – ключ шифрования должен быть доступен только тем, кому предназначено сообщение. Симметричные криптосистемы называют еще *одноключевыми* криптографическими системами, или *криптосистемами с закрытым ключом*.

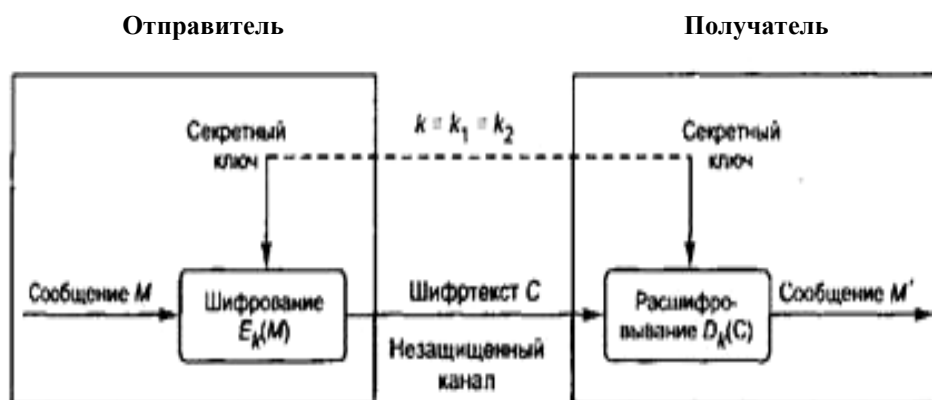


Схема симметричной криптосистемы шифрования

Данные криптосистемы характеризуются наиболее высокой скоростью шифрования, и с их помощью обеспечиваются как конфиденциальность и подлинность, так и целостность передаваемой информации. Конфиденциальность передачи информации с помощью симметричной криптосистемы зависит от надежности шифра и обеспечения конфиденциальности ключа шифрования [2].

С одной стороны, такая схема имеет те недостатки, что необходимо кроме открытого канала для передачи шифрограммы наличие также секретного канала для передачи ключа, а кроме того, при утечке информации о ключе, невозможно доказать, от кого из двух корреспондентов произошла утечка.

С другой стороны, среди шифров именно этой группы есть единственная в мире схема шифровки, обладающая абсолютной теоретической стойкостью. Все прочие можно расшифровать хотя бы в принципе. Такой схемой является обычная шифровка с ключом, длина которого равна длине сообщения. При этом ключ должен использоваться только раз. Любые попытки расшифровать такое сообщение беспо-

лезны, даже если имеется априорная информация о тексте сообщения. Осуществляя подбор ключа, можно получить в результате любое сообщение [3].

Асимметричные шифры

Асимметричные криптографические системы были разработаны в 1970-х годах. Принципиальное отличие асимметричной криптосистемы от криптосистемы симметричного шифрования состоит в том, что для шифрования информации и ее последующего расшифровывания используются различные ключи:

- *открытый ключ* K используется для шифрования информации, вычисляется из секретного ключа k ;
- *секретный ключ* k используется для расшифровывания информации, зашифрованной с помощью парного ему открытого ключа K .

Эти ключи различаются таким образом, что с помощью вычислений нельзя вывести секретный ключ k из открытого ключа K . Поэтому открытый ключ K может свободно передаваться по каналам связи.

Асимметричные системы называют также двухключевыми криптографическими системами, или криптосистемами с открытым ключом.



Схема асимметричной криптосистемы шифрования

Для криптографического закрытия и последующего расшифровывания передаваемой информации используются открытый и секретный ключи получателя B сообщения.

В качестве ключа зашифровывания должен использоваться открытый ключ получателя, а в качестве ключа расшифровывания – его секретный ключ.

Секретный и открытый ключи генерируются попарно. Секретный ключ должен оставаться у его владельца и быть надежно защищен от НСД (аналогично ключу шифрования в симметричных алгоритмах). Копия открытого ключа должна находиться у каждого абонента криптографической сети, с которым обменивается информацией владелец секретного ключа [5].

Сравнение шифров

Достоинства симметричной криптосистемы:

- скорость;
- простота реализации (за счёт более простых операций);
- меньшая требуемая длина ключа для сопоставимой стойкости;
- изученность (за счёт большего возраста);

Недостатки симметричной криптосистемы:

- сложность управления ключами в большой сети;
- сложность обмена ключами. Для применения необходимо решить проблему надёжной передачи ключей каждому абоненту, так как нужен секретный канал для передачи каждого ключа обеим сторонам [6].

Достоинства асимметричной криптосистемы:

- не нужно предварительно передавать секретный ключ по надёжному каналу;
- только одной стороне известен ключ дешифрования, который нужно держать в секрете (в симметричной криптографии такой ключ известен обеим сторонам и должен держаться в секрете обеими);
- в больших сетях число ключей в асимметричной криптосистеме значительно меньше, чем в симметричной.

Недостатки асимметричной криптосистемы:

- в алгоритм сложнее внести изменения;
- более длинные ключи – ниже приведена таблица, сопоставляющая длину ключа симметричного алгоритма с длиной ключа RSA с аналогичной криптостойкостью;
- шифрование-расшифровывание с использованием пары ключей проходит на два-три порядка медленнее, чем шифрование-расшифрование того же текста симметричным алгоритмом;
- требуются существенно большие вычислительные ресурсы, поэтому на практике асимметричные криптосистемы используются в сочетании с другими алгоритмами [5].

ЛИТЕРАТУРА

1. Шифр [Электронный источник]. – 2017. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80>. – Дата доступа: 24.09.2017.
2. Симметричные криптосистемы шифрования [Электронный источник]. – 2017. – Режим доступа: <http://yupn.ru/193/symmetric-encryption-system>. – Дата доступа: 24.09.2017.
3. Типы шифров [Электронный источник]. – 2017. – Режим доступа: http://studbooks.net/2243124/informatika/typy_shifrov. – Дата доступа: 24.09.2017.
4. Асимметричные криптосистемы шифрования [Электронный источник]. – 2017. – Режим доступа: <http://yupn.ru/197/asymmetric-encryption-system/>. – Дата доступа: 24.09.2017.
5. Криптосистема с открытым ключом [Электронный источник]. – 2017. – Режим доступа: https://ru.wikipedia.org/wiki/Криптосистема_с_открытым_ключом. – Дата доступа: 24.09.2017.
6. Симметричные криптосистемы [Электронный источник]. – 2017. – Режим доступа: https://ru.wikipedia.org/wiki/Симметричные_криптосистемы. – Дата доступа: 24.09.2017.

УДК 004.056.55

ПРОБЛЕМАТИКА ПОПУЛЯРНЫХ АЛГОРИТМОВ ШИФРОВАНИЯ

Р.Ю. КАРАБАНОВ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматриваются самые популярные алгоритмы шифрования и их проблемы в современном информационном мире. Приводятся примеры проблем алгоритмов, в частности опыт взлома и их анализ.

В современном мире приводятся множество требований к алгоритму шифрования, начиная от криптостойкости и заканчивая временем для шифрования/дешифрования данных. Однако самым главным критерием оценки проблемы будем считать факт взлома алгоритма до этого. В данной работе остановимся на трех самых популярных на данное время алгоритмах – AES, DES, RSA. Именно эти три алгоритма реализованы в стандартной библиотеке «*javax.crypto*» в языке программирования Java – самом популярном объектно-ориентированном языке программирования на 2016 год [1]. Также следует отметить, что данные алгоритмы были введены в общее пользование до нынешнего десятилетия [2–4], а следовательно, не могут, в эру быстро развивающихся информационных технологий называться современными.

1. **DES** (англ. *data encryption standard*) – алгоритм для симметричного шифрования, разработанный фирмой IBM и утвержденный правительством США в 1977 году как официальный стандарт (FIPS 46-3). Размер блока для DES равен 64 бита. В основе алгоритма лежит сеть Фейстеля с 16 циклами (раундами) и ключом, имеющим длину 56 бит. Алгоритм использует комбинацию нелинейных (S-блоки) и линейных (перестановки E, IP, IP-1) преобразований [4].

Основная проблема. Так как длина ключа равна 56 битам, существует 2^{56} возможных ключей. Сегодня такая длина ключа недостаточна, поскольку допускает успешное применение лобовых атак. Альтернативой DES можно считать тройной DES, IDEA, а также алгоритм Rijndael, принятый в качестве нового стандарта на алгоритмы симметричного шифрования.

Также без ответа пока остается вопрос, возможен ли криптоанализ с использованием существующих характеристик алгоритма DES. Основой алгоритма являются восемь таблиц подстановки, или S-boxes, которые применяются в каждой итерации. Имеет место опасность, что эти S-boxes конструировались таким образом, что криптоанализ возможен для взломщика, который знает слабые места S-boxes. В течение многих лет обсуждалось как стандартное, так и неожиданное поведение S-boxes, но все-таки никому не удалось обнаружить их фатально слабые места [5].

Вывод: данный алгоритм можно легко взломать с помощью перебора.

2. **Advanced Encryption Standard (AES)** – симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством США по результатам конкурса AES. Этот алгоритм хорошо проанализирован и сейчас широко используется, как это было с его предшественником DES. Национальный институт стандартов и технологий США (англ. *National Institute of Standards and Technology, NIST*) опубликовал спецификацию AES 26 ноября 2001 года после пятилетнего периода, в ходе которого были созданы и оценены 15 кандидатур. 26 мая 2002 года AES был объявлен стандартом шифрования [3]. Суть AES в том, что любая «лобовая атака» на защищенные данные, то есть подбор всех возможных паролей, в перспективе очень сильно растягивается. Если представить, что взломщик располагает огромными ресурсами, то есть целой коллекцией суперкомпьютеров, то при усердном старании доступ к зашифрованным данным он мог бы получить через десятки лет. Если же в его распоряжении ничего этого нет, то взлом AES займет астрономически долгое время [6].

Основная проблема. Криптографы нашли метод взлома ключей AES-шифра ещё в 2011 году во время конференции по криптографии *Crypto 2011*, которая проходила в Санта Барбаре, США. Данный метод позволяет злоумышленникам получать секретные AES ключи в пять раз быстрее, чем это делалось ранее. Он же использует технику двудольного криптоанализа для удаления двух битов из 128-, 192- и 256-битных ключей. И эксперты поспешили успокоить общественность: «Данное исследование пошатнуло многие основы, так как мы получили первый метод взлома, AES-шифра с одним ключом, (немного) более быстрый, чем брут форс атака, – заявил Нейт Лаусон, криптограф и начальник отдела консультаций по безопасности компании Root Labs. – Однако с практической точки зрения, AES-шифр взломать пока невозможно» [7].

Тем не менее уже меньше, чем через год, Сергей Скоробогатов, который до этого недавно обнаружил бэкдор в китайском чипе FPGA, произвел успешное извлечение ключа AES, который отмечен как «высокозащищенный» и «практически неломаемый», на FPGA военного уровня Actel/Microsemi ProASIC3 за 0,01 секунду. Использовался способ анализа сторонних каналов, который называется Pipe-

line Emission Analysis (PEA). Это не новый способ сам по себе, а значительно улучшенный метод волнового анализа. Данный способ используется для атаки на реализацию AES на FPGA, используя наводки и помехи [8].

Вывод: данный алгоритм оказался тоже взламываемым при помощи метода волнового анализа. Также существует вероятность, что при помощи квантовых компьютеров в будущем это будет сделать достаточно быстро, чем это предполагали, когда создавали алгоритм.

3. **RSA** (аббревиатура от фамилий Rivest, Shamir и Adleman) – криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.

Криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи. Алгоритм используется в большом числе криптографических приложений, включая PGP, S/MIME, TLS/SSL, IPSEC/IKE и других [2].

Основная проблема. Была обнаружена уязвимость в реализации алгоритма RSA, позволившая исследователям взломать 1024-битное шифрование. Но для реализации взлома необходим физический доступ к держателю «секретного ключа», так что на крупных компаниях это вряд ли отразится.

Что же касается обычных гаджетов, то достаточно не упускать их из виду. Докторант Мичиганского университета Андреа Пеллегрини (Andrea Pellegrini) завтра представит отчет о продланной работе на конференции «Design, Automation and Test in Europe» (DATE) в Дрездене.

Но перейдем к сути. Атака на алгоритм проводилась путем искусственного вызова ошибок при помощи изменения напряжения на процессоре. В результате появлялись ошибки в коммуникации с другими клиентами, и удавалось получить небольшую часть ключа, а как только было собрано достаточно частей, ключ был восстановлен в режиме офлайн. На все про все ушло 104 часа работы 81 процессора Pentium 4. Техника не пострадала, следов взлома не осталось. Несмотря на то, что в статье описывается только уязвимость, ученые из университета заявили, что они предлагают довольно простое решение проблемы. Для этого, по их словам, достаточно применить «соль», позволяющую изменять порядок цифр случайным образом при каждом запросе ключа [9].

Таким образом, RSA также был взломан, а алгоритм до сих пор не изменен, следовательно, если бы ученые стремились заново его взломать, то потратили даже меньше 104 часов, потому что вычислительная мощность компьютеров увеличилась.

Заключение

Распространенность алгоритмов шифрования часто служит им плохую службу, потому что все больше злоумышленников и не только пытаются их взломать. Однако стоит отметить, что для своего времени каждый был прорывом в шифровании. И новое время требует новое откровение в среде шифрования данных.

ЛИТЕРАТУРА

7. The 9 Most In-Demand Programming Languages of 2016 [Electronic resource]. – 2016. – Mode of access: <http://www.codingdojo.com/blog/9-most-in-demand-programming-languages-of-2016/>. – Date of access: 23.09.2017.
8. RSA [Электронный ресурс]. – 2017. – Режим доступа: <https://ru.wikipedia.org/wiki/RSA/>. – Дата доступа: 23.09.2017.
9. AdvancedEncryptionStandard [Электронный ресурс]. – 2017. – Режим доступа: https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard. – Дата доступа: 23.09.2017.
10. DES [Электронный источник]. – 2017. – Режим доступа: <https://ru.wikipedia.org/wiki/DES>. – Дата доступа: 23.09.2017.
11. Криптографические основы безопасности [Электронный ресурс]. – 2017. – Режим доступа: <http://www.intuit.ru/studies/courses/28/28/lecture/20412?page=4#sect17>. – Дата доступа: 23.09.2017.
12. Алгоритм шифрования AES [Электронный ресурс]. – 2017. – Режим доступа: <https://www.opengsm.ru/blog/algorithm-shifrovaniya-aes/>. – Дата доступа: 23.09.2017.
13. Криптографы нашли способ взлома ключей AES-шифра [Электронный ресурс]. – 2011. – Режим доступа: <http://www.securitylab.ru/news/406880.php>. – Дата доступа: 23.09.2017.
14. Извлечение AES ключа в мгновение ока [Электронный ресурс]. – 2012. – Режим доступа: <https://habrahabr.ru/post/145775/>. – Дата доступа: 23.09.2017.
15. Взломано 1024-битное шифрование RSA [Электронный источник]. – 2010. – Режим доступа: <https://habrahabr.ru/post/86841/>. – Дата доступа: 23.09.2017.

УДК 004.02

ОБЗОР СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ, ОСНОВАННЫХ НА MYSQL

В.С. КУПЦОВ

(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Рассматриваются плюсы и минусы систем управления базами данных, основанных на MySQL. Проанализированы возможности самых популярных форков. Рассмотрены новые хранилища данных.

Сервер реляционной базы данных MySQL в короткое время успел стать сверхпопулярной базой данных, а также незаменимой частью современного Интернета, входя в связку открытых веб-технологий LAMP (Linux-Apache-MySQL-PHP), которая и формирует технологически практически весь современный Веб. Роль баз данных в этой связке имеет исключительный характер. Поэтому архитектура современного сайта всегда содержит быстрое и гибкое хранилище данных, такое как MySQL. Также стоит отметить большое количество систем управления базами данных, основанных на MySQL.

Далее будут рассмотрены возможности самых популярных форков (ответвлений кода от основного проекта) MySQL.

ExtSQ – проект параллельно ведет две ветви разработки, базирующихся соответственно на кодовой ветке 5-й версии оригинального MySQL. ExtSQL разрабатывался с сильным уклоном для специализированного использования в системах веб-хостинга и призван решить проблемы, связанные с организацией учета потребления ресурсов. Администраторы ExtSQL получают возможность полного мониторинга активности пользователей, всех баз и соединений. Например, запрос «SHOW STATISTICS select, insert FROM user HISTORY» позволит узнать число запросов «select» и «insert», совершенных пользователями за последний час. Естественно, для этого в бывший MySQL добавлены новые команды и расширен диалект SQL, при этом сохранена полная обратная совместимость с MySQL. Плата за подобные надстройки над традиционным MySQL выражается в потере производительности сервера в среднем на 5–6% на обычных задачах, и на 12–16% при интенсивной работе с реально большими базами данных.

PerconaServer – сборка MySQL с включенным по умолчанию XtraDBstorageengine. Отличается от MySQL+InnoDBplugin лучшей производительностью и масштабируемостью, особенно на современных многоядерных серверах. Также улучшена функциональность – больше полезной для оптимизации статистики и пр. Полностью совместим с таблицами InnoDB, то есть можно переходить от InnoDB к XtraDB и обратно без проблем. Хранилище XtraDB основано на коде InnoDB-plugin, полностью совместимо с ним, но отличается заметно более высокой производительностью, благодаря интеграции патчей от компаний Google и Percona. В частности, в XtraDB улучшен механизм работы с памятью, улучшена работа подсистемы ввода/вывода InnoDB, добавлена поддержка нескольких потоков чтения и записи, поддержка управления пропускной способностью, реализация упреждающей выборки данных (read-ahead), адаптивная установка контрольных точек (adaptivecheckpointing), расширены возможности по масштабированию для больших проектов, система организации блокировок адаптирована для работы на системах с большим числом CPU, добавлены дополнительные возможности для накопления и анализа статистик.

MariaDB – сборка, синхронизированная с кодовой базой MySQL и полностью с ней совместимая, то есть может выступать в качестве прозрачной замены MySQL, обладая при этом рядом расширенных функций, включая оптимизации производительности и поставляясь с набором дополнительных движков хранилищ.

Новые хранилища данных:

- 1) **Aria** – высоконадежное хранилище, отличающееся повышенной устойчивостью сохранения целостности данных после краха, при полной совместимости с MyISAM;
- 2) **OQGRAPH** – хранилище для организации сложных графов;
- 3) **Sphinx** – хранилище для построения поисковых движков, обеспечивает высокую производительность индексации и поиска, позволяя делать запросы с помощью SphinxQL. Хорошо масштабируется, позволяя работать с миллиардами документов с возможностью распределенного поиска;
- 4) **PBXT (PrimeBase XT)**, поддерживающее мультиверсионный метод организации хранения данных MVCC (multi-versionconcurrencycontrol), позволяющий избавиться от блокировок при выполнении операций чтения. Поддерживает ACID-совместимые транзакции, быстрый откат транзакций и восстановление после некорректного завершения работы сервера. Имеются средства для обеспечения ссылочной целостности данных, поддержка определения внешних ключей, каскадных обновлений и удалений данных;
- 5) **FederatedX** – позволяет организовать обращение к удаленным таблицам как к локальным.

Другие улучшения:

- 1) патчи MyISAM движка – Сегментированный кэш, который при высоких нагрузках дает существенный прирост производительности;
- 2) виртуальные столбцы;
- 3) ликвидация таблиц – новый вид оптимизации запросов с использованием JOIN;
- 4) пул потоков – теперь на одно соединение можно открыть больше одного потока;
- 5) улучшены механизмы отладки медленных запросов.

Совершенно противоположным путем пошел другой, уже достаточно популярный и перспективный форк **MySQL** – проект Drizzle. Этот проект представляет собой упрощенный и более быстрый вариант MySQL, в котором тщательно отобраны и удалены все ресурсоемкие и маловостребованные возможности MySQL 5. Часть из этих возможностей все же возможно реализовать через подключаемые плагины. Эта СУБД позиционируется как высокоскоростная и высоконадежная БД, с поддержкой ACID-транзакций. В качестве хранилища используется InnoDB и PBXT. Исходный код MySQL, написанный на С, был полностью переписан на языке С++. Управление проектом находится в руках независимого сообщества. Архитектура Drizzle построена на максимальном упрощении структуры БД и выносе логики на сторону приложений. В частности, такой дизайн СУБД позволяет организовать обработку большого числа параллельных запросов, при выполнении которых в полной мере задействуются мощности современных многоядерных CPU. Кроме этого, в Drizzle дополнительно реализованы встроенные средства для разнесения данных по ключевому полю (sharding) на кластер из нескольких машин, для создания эффективной балансировки нагрузки для по-настоящему сверхнагруженных проектов. По сравнению с MySQL в Drizzle удалена поддержка хранимых процедур (вместо CREATE FUNCTION следует использовать связываемые объекты), триггеров, кэша запросов (querycache), представлений (view), операции GRANT и ALTER, ограничений ACL, команды SHOW, предварительно подготовленных запросов (preparedstatement) и др. Прекращена поддержка малоиспользуемых типов данных из MySQL.

OurDelta – это ответвление от MySQL отбирает самые интересные доработки от независимых разработчиков, после чего собирает, интегрирует и синхронизирует их всех в одном продукте. Чтобы дать возможность выбора, OurDelta ведет две параллельные ветви своих популярных ребилдов: на базе исходников классического MySQL и вышерассмотренного MariaDB. Чтобы иметь представление, что же включается в билды от OurDelta, остановимся на одном из ее крупных разработчиков патчей – проекте Facebook. Чтобы справиться с нагрузкой, которую создают более миллиарда пользователей социальной сети, Facebook оперирует четырьмя тысячами экземпляров MySQL (при этом очень активно используется шардинг), всего задействовано девять тысяч инсталляций memcached. Конечно, стандартный MySQL не предназначен для подобных нагрузок, поэтому уже более 5 лет идет интенсивная доработка MySQL под собственные экстремальные потребности Facebook. Неудивительно, что patchset от Facebook – один из самых интересных в составе OurDelta.

В заключение можно сделать **вывод**, что выбирать систему управления базами данных следует в зависимости от размеров, потребностей проекта и нагрузки на нем.

ЛИТЕРАТУРА

1. Bloggerator'snewslne – новостной HTTP-блог об ИТ [Электронный ресурс] / MySQL«на стероидах» – Режим доступа: <http://bloggerator.org/page/mysql-na-steroidah>. – Дата доступа: 20.09.2017.
2. Хабрахабр [Электронный ресурс] / Форки движка MySQL: MariaDB, Percona. whoiswho? – Режим доступа: <https://habrahabr.ru/post/108104/>. – Дата доступа: 20.09.2017.

УДК 004.02

**ERP-СИСТЕМА КАК ИНСТРУМЕНТ ДЛЯ СТРУКТУРИРОВАНИЯ
И АВТОМАТИЗАЦИИ ПРОЦЕССА УПРАВЛЕНИЯ ПРЕДПРИЯТИЕМ****В.С. КУПЦОВ***(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматривается понятие системы управления предприятием (ERP-системы) и цели их внедрения в работу предприятия. Показано, что внедрение системы управлением предприятия делает бизнес-процессы компании более прозрачными, повышая ее привлекательность для инвесторов. Использование данных систем целесообразно и позволяет упростить и оптимизировать работу предприятия.

ERP-системы (EnterpriseResourcePlanning, или Планирование Ресурсов Предприятия) являются удобным инструментом для современных руководителей. Системы класса ERP – это многофункциональный набор связанных между собой приложений в виде среды для автоматизации планирования, контроля, анализа и управления бизнес-процессами предприятия в сфере финансов, товарно-материальных, производственных и логистических потоков, бухгалтерского учета, управления персоналом, отношений с контрагентами.

К наиболее популярным ERP-системам на территории СНГ относятся:

- 1) 1C:ERP;
- 2) MS Dynamics ERP;
- 3) Sage ERP X3;
- 4) Ultima;
- 5) Инталев 24;
- 6) NetSuite;
- 7) КОМПАС;
- 8) Oracle E-Business Suite и др.

Системы управления ресурсами компании внедряются для того, чтобы объединить все подразделения компании и все необходимые функции в одной компьютерной системе, которая будет обслуживать текущие потребности этих подразделений. Тогда как традиционные системы используют многоступенчатую схему обработки данных, при которой за фиксацией информации на бумаге следует передача бумажных носителей в конкретное структурное подразделение, где данные переформируются или вводятся в компьютер, в системах управления предприятием помещение информации в базу данных производится непосредственно от источника, минуя промежуточные стадии, после чего данные становятся доступны всем заинтересованным лицам. Система ведет единую базу данных по всем подразделениям и задачам, так что доступ к информации становится проще, а главное, подразделения получают возможность обмениваться информацией.

ERP-система автоматизирует задачи, встроенные в выполнение бизнес-процессов. Так, при получении заказа от потребителя менеджер имеет всю информацию об отношениях с заказчиком. Когда одно подразделение заканчивает работать с заказом, тот автоматически передается в следующее подразделение. При этом исключается возможность возникновения информационных несоответствий внутри системы, что значительно повышает качество информации и обеспечивает дополнительные преимущества при принятии решений. В итоге заказы обрабатываются быстрее и без ошибок. Аналогичные возможности возникают у многих других служб – службы персонала, производственного отдела, отдела маркетинга, службы снабжения. Единая информационная база позволяет учитывать взаимосвязь отдельных процессов, как, например, загрузка заказами на текущий месяц и график отпусков персонала. Благодаря широкой доступности информации отпадает необходимость в посреднических услугах по передаче данных между сотрудниками или от руководства компании к ее служащим. Данное обстоятельство, в свою очередь, позволяет сократить штат компании за счет работников, чьи функциональные обязанности ограничиваются исключительно подготовкой и распространением информации.

Система меняет отношение персонала в отдельности и служб в целом к своей работе. Теперь менеджеры, ведущие заказ, владеют информацией о его состоянии в каждый момент времени и по всем аспектам: поступила ли оплата за работу, достаточно ли комплектующих на складе, стоит ли в графике выполнения работ их заказ к нужному сроку. И если, например, склад вовремя не введет в систему информацию о запасах, то получивший запрос клиента менеджер, справившись в системе о состоянии

склада, может ответить отказом на его обращение, увидев, что запас на складе недостаточен или нужное изделие отсутствует. Тем самым система повышает ответственность каждого за общее дело.

Благодаря единовременному вводу данных в общую базу предприятия информация может использоваться одновременно любым количеством пользователей системы, что повышает эффективность как планирования, так и контроля, а также обеспечивает согласованность действий всех звеньев предприятия и продуктивность управления в целом.

ERP-системы с веб-интерфейсом делают возможным объединение географически удаленных друг от друга филиалов и отделений компании и вовлечение их в единое информационное пространство, что позволяет осуществлять обмен организационными наработками между структурными подразделениями с нестандартными и высокоэффективными процессами, а также способствует формированию единого корпоративного образа и улучшению имиджа компании на мировом рынке.

Системы управления предприятием позволяют создавать своего рода информационные каналы, способные связать предприятие с компаниями-партнерами для предоставления им доступа к базе данных предприятия и обмена необходимой информацией. В результате такого сотрудничества формируется единый информационный архив, что позволяет оптимизировать процессы снабжения, поставки и другие виды совместной деятельности, а также вывести на качественно новый уровень весь процесс взаимодействия в целом.

Внедрение системы управлением предприятия делает бизнес-процессы компании более прозрачными, повышая ее привлекательность для инвесторов, а следовательно, позволяет привлечь дополнительные инвестиции для дальнейшего развития производства.

Внедрение такой системы, как правило, является длительным и сложным процессом. У каждого крупного производителя ERP системы разработаны свои технологии и методики внедрения. Эти методики несколько отличаются друг от друга, но общий порядок действий, в основном, одинаковый. Кроме того, порядок внедрения может различаться в зависимости от программной и аппаратной архитектуры системы. На эти методики стоит обращать внимание, когда осуществляется выбор поставщика услуг.

Организации, которая приняла решение внедрять ERP систему, необходимо со своей стороны предпринять ряд действий для успешной реализации проекта внедрения.

К таким действиям относятся:

1. *Подготовка внедрения.* На этапе подготовки организации необходимо определить основные цели, ради чего будет внедряться ERP система. Это позволит четко понять результаты и ожидания от внедрения. Так как внедрение системы является довольно затратным проектом, необходимо оценить предполагаемый бюджет, который организация сможет выделить на внедрение. Также, на этом этапе определяется ответственный за проект, и выявляются ключевые специалисты (члены команды проекта со стороны организации), определяется порядок их взаимодействия.

2. *Анализ бизнеса.* Эти действия являются одними из самых критических во всем проекте. Как правило, ERP система должна работать в течение десятка лет и более. Поэтому необходимо определить перспективы развития как самой организации, так и рынка на длительный период.

3. *Выбор системы управления ресурсами предприятия.* Выбор системы необходимо осуществлять на основе текущих и будущих потребностей бизнеса.

4. *Выбор поставщика.* Как правило, на рынке существует много поставщиков одной и той же ERP системы. Это компании вендоры или системные интеграторы. Они предлагают комплекс услуг по техническому внедрению системы.

5. *Управление проектом.* Ответственный за внедрение ERP системы и специалисты организации должны осуществлять управление проектом внутри компании. Они должны вести план проекта, контролировать сроки, бюджет, объем работ, соответствие целям внедрения.

6. *Тестирование.* Даже при самой лучшей организации внедрения, существует риск возникновения ошибок в работе системы. Поэтому по ходу ввода в эксплуатацию функционала ERP системы необходимо предусмотреть обязательное тестирование работы процессов, подразделений и модулей. Лучшим способом тестирования является выполнение параллельной работы в старой системе и во внедряемой системе. Это позволит избавиться от основных ошибок.

7. *Тренинги и обучение.* Обучение сотрудников является обязательным условием работы системы. В зависимости от сложности системы оно может занять несколько недель. Перед началом обучения необходимо убедиться, что ERP система была протестирована. Одним из неудачных вариантов является случай, когда поставщик услуг совмещает тестирование и обучение пользователей. В этом варианте сотрудники организации (будущие пользователи системы) выступают в качестве тестировщиков системы.

8. *Ввод в эксплуатацию.* Ввод в эксплуатацию является важным моментом в процессе внедрения ERP системы. Существует два варианта ввода системы в эксплуатацию:

- 1) запуск системы сразу во всей организации;
- 2) поэтапный ввод.

Второй вариант является более предпочтительным, так как позволяет постепенно переходить на новые условия работы. В случае возникновения ошибок или проблем в работе, будет затронута только часть бизнеса (отдельные процессы или подразделения), а не вся организация.

В заключение можно сделать *вывод*, что, несмотря на довольно высокую стоимость ERP-систем и некоторые сложности при внедрении, использование данных систем целесообразно и позволяет упростить и оптимизировать работу предприятия.

ЛИТЕРАТУРА

1. ЦМД-софт [Электронный ресурс] / ERP | Что такое ERP-система? – Режим доступа: http://www.cmdsoft.ru/information_systems/erp/what_is_erp_system/. – Дата доступа: 20.09.2017.
2. Онлайн бухгалтерия и ERP [Электронный ресурс] / Топ 10: Онлайн ERP системы. – Режим доступа: <http://www.clouderp.ru/tools/erp/>. – Дата доступа 20.09.2017.
3. Хабрахабр [Электронный ресурс] / Что такое ERP система. – Режим доступа: <https://habrahabr.ru/post/333018/>. – Дата доступа: 20.09.2017.

УДК 004.042

ПРОБЛЕМЫ ПАРАЛЛЕЛИЗМА В ПРОГРАММИРОВАНИИ

Ю.В. ЛАПТЕВ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматриваются некоторые принципиальные проблемы обеспечения параллельной работы программных приложений и способы их решения.

Параллельное выполнение операций – одна из наиболее сложных дисциплин в области разработки программного обеспечения. Проблемы параллелизма возникают каждый раз, когда несколько процессов или потоков вычислений предпринимают попытки манипуляций одними и теми же данными. Также параллельные операции трудно тестировать и невозможно перечислить все возможные сценарии развития событий, способным привести к тем или иным неприятным последствиям [1].

Проблемы параллелизма

Рассмотрим некоторые проблемы обеспечения параллельной работы приложений, на преодоление которых направлены усилия систем управления параллельными заданиями.

Рассмотрим *пример утраты изменений*. Предположим, что разработчик А открывает на редактирование файл Customer, с исходным кодом, и правит в нем какой-то метод. В это же время разработчик В обращается к этому же файлу и правит тот же метод. Делает он это быстрее первого разработчика и сохраняет изменения. Разработчик А об этом ничего не знает, и как итог, сохраняя свои изменения, перезапишет все изменения второго разработчика. Данная проблема выражается в потере достоверности информации и в некорректном поведении системы, чего можно было бы избежать, если бы два человека не обращались к одним и тем же данным одновременно. Естественно можно упорядочить действия таким образом, чтобы только один субъект имел право обращаться к данным в определенный момент времени, но это снизит возможность параллельной обработки информации.

Также имеется одна из весьма важных и трудных проблем, связанная с взаимоблокировками. Предположим, что разработчик А приступает к редактированию файла Customer, а разработчик В редактирует файл Order. В какой-то момент времени разработчик В понимает, что для завершения работы ему необходимо несколько изменить и файл Customer, но разработчик А владеет блокировкой на этот файл и второму разработчику приходится ждать ее освобождения. В то же время первому разработчику приходит мысль, что нужно подправить файл Order, но этому мешает блокировка, удерживаемая разработчиком В. Оба разработчика попадают в ситуацию взаимоблокировок, и ни один не может завершить свою работу, пока другой разработчик не завершит свою. В данном случае проблема не выглядит угрожающе и ее несложно предотвратить, но если цепочку взаимозависимостей составляет множество людей или программ, тогда и начинаются серьезные проблемы.

Рассмотрим еще одну проблему с бесконечной отсрочкой, когда одна или несколько задач должны ожидать до тех пор, пока не произойдет какое-то событие или не создадутся определенные условия. Предположим, что если установить для электронного банка правила, предписывающие всем задачам снятия денег со счета находиться в состоянии ожидания до тех пор, пока не будут выполнены все задачи вложения денег на счет, то задачи снятия денег рискуют стать бесконечно отсроченными. Если будут без конца поступать запросы на пополнение одного и того же счета, то такая ситуация может вызвать бесконечную отсрочку задач снятия денег [2].

Транзакции

Транзакции являются основным инструментом управления параллельными процессами в приложениях. Транзакция представляет собой ограниченную последовательность действий с явно определенными начальной и завершающей операциями. Например, транзакция, связанная с банкоматом, начинается с размещения магнитной карты в считывающем устройстве и завершается выдачей денег или сообщением о несоответствии запрошенной суммы остатку на счете. Все ресурсы, затрагиваемые транзакцией, пребывают в согласованном состоянии в момент ее начала и остаются в таковом после ее завершения. Кроме этого, транзакция должна либо выполняться целиком, либо не выполняться вовсе. Банковская система не может вычесть сумму остатка на счете до тех пор, пока клиенту не будет выдана соответствующая пачка купюр. Транзакции часто описывают в терминах свойств, которые обозначаются аббревиатурой ACID:

- **atomicity** (атомарность). В контексте транзакции либо выполняются все действия, либо не выполняется ни одно из них. Частичное или избирательное выполнение недопустимо. Например, если клиент банка переводит сумму с одного счета на другой и в момент между завершением расходной и началом приходной операции сервер терпит крах, система должна вести себя так, будто расходной операции

не было вовсе. Система должна либо осуществить обе операции, либо не выполнить ни одной. Фиксация результатов служит свидетельством успешного окончания транзакции, а откат приводит систему в состояние, в котором она пребывала до начала транзакции;

- **consistency** (согласованность). Системные ресурсы должны пребывать в целостном и непротиворечивом состоянии как до начала транзакции, так и после ее окончания;

- **isolation** (изолированность). Промежуточные результаты транзакции должны быть закрыты для доступа со стороны любой другой действующей транзакции до момента их фиксации. Иными словами, транзакция протекает так, будто в тот же период времени других параллельных транзакций не существует;

- **durability** (устойчивость). Результат выполнения завершенной транзакции не должен быть утрачен ни при каких условиях.

В большинстве случаев под транзакциями в приложениях подразумеваются последовательности операций, описывающие процессы взаимодействия с базами данных. Но существует и множество других объектов, управляемых с помощью механизмов поддержки транзакций: например, очереди сообщений или заданий на печать, банкоматы и т.д. Для обеспечения высокого уровня пропускной способности современные системы управления транзакциями проектируются в расчете на максимально короткие транзакции. Обычно из практики исключаются транзакции, которые охватывают действия по обработке нескольких запросов. Если же подобной ситуации избежать не удастся, решения реализуются на основе схемы длинных транзакций. Чаше границы транзакции совпадают с моментами начала и завершения обработки одного запроса. Подобная транзакция запроса – весьма удачная модель, и многие среды поддерживают простой и естественный синтаксис ее описания. Альтернативное решение состоит в том, чтобы как можно дольше откладывать процедуру открытия транзакции. При использовании подобной отсроченной транзакции все операции чтения могут выполняться до момента ее начала, который наступает только при необходимости осуществления операций, связанных с внесением каких-либо изменений. Это позволяет минимизировать длину транзакции, но лишает возможности на протяжении продолжительных периодов времени применять какие-либо средства управления параллельными операциями. Такая стратегия сопряжена с повышением вероятности возникновения эффекта несогласованного чтения и потому используется сравнительно редко [3].

Применяя транзакции, следует понимать, что именно надлежит блокировать. Во многих случаях диспетчер транзакций СУБД блокирует отдельные записи таблицы базы данных, вовлеченные в операцию, что позволяет сохранить высокий уровень параллелизма при доступе к таблице. Но если транзакция пытается блокировать слишком большое количество записей таблицы, число запросов на блокировку превышает допустимый лимит и система распространяет действие блокировки на таблицу в целом, приостанавливая выполнение конкурирующих транзакций.

Заключение

Параллелизм в программировании – достаточно обширная и сложная тема. Модель транзакций позволяет избежать массы трудностей, но это не значит, что проблемами управления параллельными заданиями можно полностью пренебречь, так как многие аспекты взаимодействия приложения и системы нельзя свести к контексту единой транзакции. Во многих случаях приходится иметь дело с данными, модифицируемыми несколькими транзакциями. К тому же существуют более сложные системы с поддержкой множества одновременно протекающих потоков вычислений, где приходится применять более сложные подходы к решению проблем параллелизма.

ЛИТЕРАТУРА

1. Параллелизм [Электронный источник]. – 2017. – Режим доступа: [https://ru.wikipedia.org/wiki/Параллелизм_\(информатика\)](https://ru.wikipedia.org/wiki/Параллелизм_(информатика)). – Дата доступа: 27.09.2017.
2. Проблемы параллельного и распределенного программирования [Электронный источник]. – 2017. – Режим доступа: http://www.k2x2.info/kompyutery_i_internet/parallelnoe_i_raspredelennoe_programmirovanie_na_s/p4.php. – Дата доступа: 27.09.2017.
3. Фаулер, М. Архитектура корпоративных программных приложений : пер. с англ. / Мартин Фаулер. – М. : Издат. дом «Вильямс», 2006.

УДК 004.62

ПРИНЦИПЫ И ПРИЕМЫ ОБРАБОТКИ ОЧЕРЕДЕЙ

Ю.В. ЛАПТЕВ*(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)*

Рассматривается понятие очередей, их применение в высоконагруженных проектах и принципы обработки. Показаны области применения очередей, инфраструктура обработки очередей.

Очередью в программировании называется упорядоченный набор элементов, которые могут удаляться с ее начала и помещаться в ее конец. Очередь в программировании используется, как и в реальной жизни, когда нужно последовательно совершить какие-то действия в порядке их поступления. Примером может служить организация событий в Windows. Когда пользователь оказывает какое-то действие на приложение, то в приложении не вызывается соответствующая процедура (в этот момент приложение может совершать другие действия), а ему присылается сообщение, содержащее информацию о совершенном действии, это сообщение ставится в очередь, и только когда будут обработаны сообщения, пришедшие ранее, приложение выполнит необходимое действие [1].

Области применения очередей

Самый распространенный способ применения – для асинхронного выполнения операций, порождаемых в процессе обработки запросов пользователя, результаты исполнения которых не требуются немедленно для осуществления обратной связи. Например, отправка почты и уведомлений, предложений дружбы, обновление поисковых индексов, сброс кэшей, постобработка данных, полученных от пользователя (перекодирование видео, нарезка фотографий), отправка данных подсистемам, время реакции/надежность которых не гарантируется. Обладая удобной инфраструктурой для распределенной обработки заданий, спектр применения очередей является еще больше:

- *отложенная обработка действий пользователя.* Не лучшая идея заставлять пользователя ждать, пока будут сохранены его данные в SQL базу или другое хранилище (а зачастую необходимо вносить изменения сразу в несколько систем). Иногда данные могут вовсе не попасть в хранилище, если клиент оборвал соединение, не дождавись ответа. Хорошей практикой является добавление события о пользовательских действиях в достаточно быстрый сервер очередей, после чего можно отвечать клиенту, что операция прошла успешно. Всю остальную работу надежно и эффективно выполнят обработчики очереди;

- *рассылка сообщений, писем и похожие операции.* Если необходимо отослать большое количество данных, при этом не перегрузив хранилища всплеском запросов, то это делается с помощью варьирования количества обработчиков очереди, размазывая пиковую нагрузку до разумного уровня, чтобы время обработки клиентских запросов не ухудшилось. И самое главное – с помощью очередей легко избежать дублирования сообщений. Крайне неприятно получить два письма об одном и том же событии;

- *транспорт для «надежной» статистики.* Передача важных (все, что связано с деньгами) данных на агрегаторы статистики. Системы агрегации статистики обычно требовательны к ресурсам процессора, и при обработке данных могут не обеспечивать необходимое для frontend-серверов время ответа. Еще одна особенность подобных систем – неравномерная загрузка, обычно связанная с обработкой данных порциями. Передача статистики через серверы очередей позволит избежать проблем с нестабильной задержкой и при этом сохранит гарантию доставки;

- *группировка событий.* Если группа событий будет обращаться к одному и тому же набору данных в других системах, имеет смысл ставить одинаковое время активации. Физический смысл ухищрений в более эффективном использовании кэшей хранилища, в которое пойдут запросы из обработчиков сообщений;

- *каскадные очереди.* Организация конечного автомата из нескольких очередей путем переключения данных между очередями по завершении очередного этапа обработки. Часто необходимо в процессе обработки сообщения выполнить ряд действий, сильно различающихся по необходимым ресурсам. В таком случае разнесение «быстрых» и «медленных» действий по разным этапам позволяет эффективно управлять необходимым количеством ресурсов, варьируя число обработчиков для каждой очереди [2].

Инфраструктура обработки очередей

По мере увеличения количества типов очередей уследить за возрастающим количеством скриптов и демонов (сноп) становится все сложнее. Нужно следить за нагрузкой, запускать достаточное количество процессов, не забывать мониторить их, а при выпадении серверов своевременно поднимать процессы на других машинах. Для решения всех этих проблем нужна инфраструктура, которой нужно предоставить

кластер серверов-обработчиков и серверы очередей, а она сама уже будет управлять всем этим. При проектировании инфраструктуры для обработки очередей необходимо учитывать следующие характеристики:

- **распределенность** – задания одного типа могут выполняться на разных серверах;
 - **отказоустойчивость** – выпадение 10-20 процентов серверов кластера не должно приводить к деградации важных очередей;
 - **гомогенность** – отсутствие какой-либо специализации серверов, из-за чего любое задание может выполняться на любом сервере;
 - **приоритизация** – есть критичные очереди, а есть те, которые могут и подождать;
 - **автоматизация** – минимум ручного вмешательства: балансировка – перераспределение обработчиков каждого типа в зависимости от потребностей; адаптация к серверу – определение количества обработчиков на каждом сервере в зависимости от его характеристик, отслеживание потребления памяти и процессора;
 - **пакетная обработка** – задания одного типа должны группироваться для возможности применения оптимизаций;
 - **статистика** – сбор данных о поведении очередей, потоке заданий, количестве, наличии ошибок;
 - **мониторинг** – оповещение о нештатных ситуациях и изменении стандартного поведения очередей.
- Схема инфраструктуры обработки очередей представлена на рисунке 1.

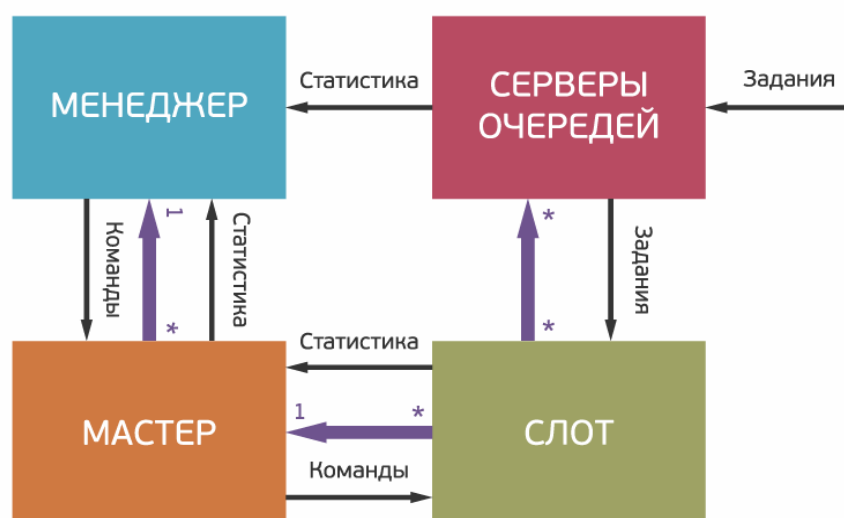


Рисунок 1. – Схема инфраструктуры обработки очередей

На данной схеме можно выделить 3 базовые сущности: слот, мастер и менеджер.

Слот – это процесс-обработчик очереди (воркер). Слотов на каждом сервере может быть запущено множество, каждый слот характеризуется типом очереди, которую он в данный момент обрабатывает. Специализация слотов на очередях определенных типов повышает локальность соединений к базам данных и внешним сервисам, а также упрощает отладку.

Мастер представляет собою процесс, ответственный за работу слотов на определенном сервере. В его обязанности входит запуск и остановка слотов, поддержание оптимального количества слотов на сервере в зависимости от его конфигурации, доступной памяти, а также мониторинг жизнедеятельности слотов, уничтожение зависших и использовавших слишком много системных ресурсов слотов.

Сущность *менеджер* выступает в роли процесса-руководителя кластера, который запускается в одном экземпляре и отвечает за весь кластер. Данный процесс на основе данных от сервера очередей и статистических данных от мастеров определяет необходимое количество слотов каждого типа и решает, какой конкретный слот будет заниматься заданиями какого типа [3].

ЛИТЕРАТУРА

1. Структуры данных, Очередь [Электронный источник]. – 2017. – Режим доступа: <https://prog-cpp.ru/data-queue/>. – Дата доступа: 27.09.2017.
2. Сервер очередей [Электронный источник]. – 2017. – Режим доступа: <https://habrahabr.ru/company/mailru/blog/216363/>. – Дата доступа: 27.09.2017.
3. Инфраструктура обработки очередей с социальной сети Мой Мир [Электронный источник]. – 2017. – Режим доступа: <https://habrahabr.ru/company/mailru/blog/228131/>. – Дата доступа: 27.09.2017.

УДК 004.021

ПРОЕКТИРОВАНИЕ REALTIME ВЕБ-ПРИЛОЖЕНИЙ ПРИ ПОМОЩИ WEBSOCKET И СЕРВЕРНЫХ СОБЫТИЙ

А.А. ЛЕВИЦКИЙ

(Представлено: Т.С. РУДЬКОВА)

Представлено теоретическое объяснение работы WebSocket-протокола, а также практический способ создания realtime веб-приложений при помощи данной технологии. Также содержится историческая справка о становлении данной технологии стандартом.

Когда данные на сервере меняются, пользователям необходимо дать об этом знать без необходимости действий с их стороны. Это один из видов увеличения производительности, который избавляет пользователя от однотипных действий, таких как обновление страницы.

Ранее приходилось эмулировать серверные события, используя для этого метод long-polling, заставляющий пользователей делать «длинные» запросы, которые оставались открытыми до того, как сервер будет готов отправить какое-либо сообщение в ответ. После получения сообщения, запрос будет закрыт, а новый – сформирован. Другое решение этой проблемы – <iframe> хаки и Flash, но данные методы не идеальны, так как требуют реализации нестандартного поведения технологий, которые не предназначены для задач, связанных с серверными событиями, а также неудобны в использовании.

Принцип работы WebSocket и алгоритмы его реализации. В 2006 году компания Opera представила серверные события (SSE) из спецификаций WHATWG Web Applications 1.0. Эта технология позволяла потоково передавать события непрерывно из веб-сервера прямо в клиентский браузер. Другие браузеры последовали этому примеру и стали реализовывать этот стандарт в 2011 году как часть спецификаций HTML5. В этом же году WebSocket-протокол был стандартизирован [2].

WebSocket позволяет открыть двустороннее соединение между клиентом и сервером, предоставляя возможность пересылать сообщения клиенту, когда какие-либо данные на сервере меняются, без необходимости клиента запрашивать эти самые данные. Это самая важная вещь для отзывчивости приложения с большим количеством одновременных соединений, например, онлайн-игры или мессенджеры. Однако до появления библиотеки socket.io, вышедшей в 2014 году, это было затруднительно [1]. Впоследствии эта библиотека стала стандартом использования WebSocket в современных веб-приложениях.

Конструктор WebSocket инициализирует соединение с сервером через протокол ws или wss (secure – безопасный). Этот протокол поддерживают все современные браузеры как на компьютерах, так и на смартфонах. Таблица поддержки браузеров проиллюстрирована на рисунке 1.

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
			59			10.2		4.4	
	14	54	60	10.1		10.3		4.4.4	
11	15	55	61	11	47	11	all	56	61
	16	56	62	TP	48				
		57	63		49				
		58	64						

Рисунок 1. – Поддержка браузеров WebSocket

Однако такая технология, как WebSocket, требует безупречной отладки, что невозможно достичь без специальных инструментов. Например, Google Chrome предоставляет возможность наблюдать сообщения при помощи вкладки Frame (рисунок 2).

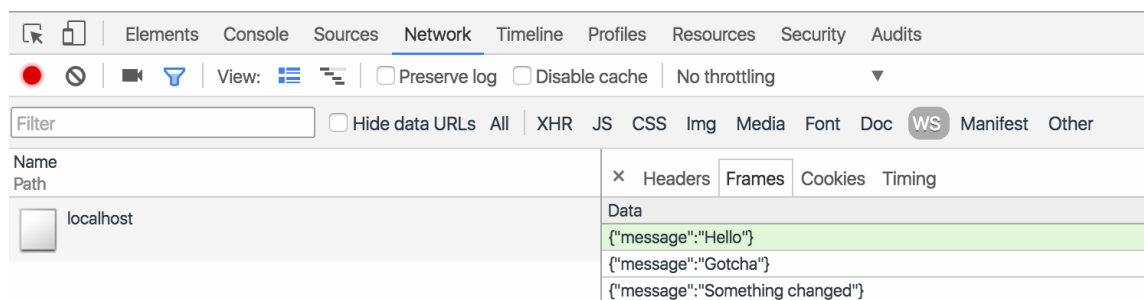


Рисунок 2. – Отладка WebSocket в Chrome

Реализация WebSocket должна присутствовать как на клиентской стороне (по средствам браузерного JavaScript или других языков, транслирующихся в JavaScript, например, Dart), так и на серверной стороне. Наиболее приспособленный для этого серверный язык – NodeJS, в котором реализация WS является встроенной. Примеры кода сообщений на клиентской и серверной части WebSocket приведены в таблице 1.

Таблица 1. – Пример сообщений клиентской и серверной части WebSocket

Client	Server
<pre>let socket = new WebSocket('ws://localhost:8081/'); socket.onopen = function() { let json = JSON.stringify({ message:'Hello ' }); socket.send(json); } socket.onmessage = function(event) { console.log(event.data); } socket.onerror = function(event){ console.log(event); } socket.onclose = function(code, reason) { console.log(code, reason); } window.addEventListener('beforeunload', function() { socket.close();});</pre>	<pre>let WSS = require('ws').Server; let wss = new WSS({ port: 8081 }); wss.on('connection', function(socket) { console.log('Opened'); let json = JSON.stringify({ message: 'Gotcha' }); socket.send(json); socket.on('message', function(message) { console.log('Received: '); }); socket.on('close', function() { console.log('Closed '); }); });</pre>

Заключение

Рассмотренный протокол WebSocket, на основе которого строятся современные realtime веб-приложения, позволяет передавать сообщения в любом количестве без необходимости клиентской части приложения запрашивать эти данные. На основе этой технологии удобно передавать сообщения для чатов, а также другие чувствительные данные по зашифрованному каналу, что не могут конкурирующие технологии. Поэтому для повышения скорости и удобства корпоративной переписки необходимо разработать программный продукт, который будет включать в себя реализацию данной технологии. Данная тема является актуальной, так как комплексного решения этой задачи в настоящее время не существует.

ЛИТЕРАТУРА

1. WebSocket [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/WebSocket>. – Дата доступа: 17.09.2017.
2. WebSockets Web API [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. – Дата доступа: 18.09.2017.

УДК 004.021

РАЗРАБОТКА КОМПЛЕКСНОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПРОЕКТАМИ ИТ-КОМПАНИЙ

А.А. ЛЕВИЦКИЙ

(Представлено: Т.С. РУДЬКОВА)

Предлагается разработка программного обеспечения, созданного для управления проектами ИТ-компаний, которая позволит компаниям управлять своими сотрудниками, а также регулировать задания заказчиков. Анализируются задачи; показан принцип работы и алгоритмы реализации.

При больших темпах роста информационных технологий появляются все больше компаний, которые недобросовестно относятся к своей работе, а также к заказчикам. Заказчики начали тщательнее выбирать исполнителей, вследствие чего молодым компаниям стало сложнее на рынке. Поэтому для более прозрачной работы появилась необходимость в отслеживании работы непосредственно заказчиком. Для более быстрого решения проблем и вопросов, стоящих перед исполнителями и заказчиками, появилась необходимость в чате между каждым из участников проекта.

Так как справляться с наплывами заказов и постоянного роста компаний обычными средствами стало сложнее, то появилась необходимость в разработке централизованной системы для организации слаженной работы команд над разработкой проектов, а также для взаимодействия с заказчиками. Она позволит оценить вклад конкретного разработчика в проект, а также с большим удобством работать исполнителям на удаленной основе.

Анализ задачи. Основным преимуществом сервиса-конкурента «Trello» можно назвать просмотр нескольких одновременно запущенных проектов и их состояние в данный момент. Для работы с группой разработчиков или иных исполнителей, работающих над проектами с конечной датой выполнения или целью, данная система может дать вам представление о ходе выполнения проектов.

Особенностью разработанной системы является то, что каждый пользователь может быстро связаться с другим пользователем посредством чата в реальном времени без необходимости писать непосредственно на корпоративную почту. Например, если заказчик захочет обсудить какие-либо правки, но сделать это нужно очень быстро вследствие занятости, то чат подойдет как нельзя лучше для таких целей. Также модуль позволит разграничить обязанности каждого пользователя.

Выбор языка программирования – один из важнейших аспектов любого проекта. NodeJS – серверная реализация языка программирования JavaScript, основанная на движке V8; предназначена для создания масштабируемых распределенных сетевых приложений, таких как веб-сервер. NodeJS по целям использования сходен с каркасами Twisted на языке Python и EventMachine на Ruby. В отличие от большинства программ JavaScript, этот каркас выполняется не в браузере клиента, а на стороне сервера. Так как NodeJS использует JavaScript синтаксис, то разработчику веб-сайтов легче переключаться между написанием клиентской части, написанной на JavaScript, и серверной частью, написанной на NodeJS. NodeJS является асинхронным языком программирования, построенный на модели событий, что позволяет выполнять код все зависимости от порядка вызовов функций. NodeJS выполняется в одном потоке, но асинхронность достигается путем подписки на события и вызова кластеров [1].

Выбор СУБД пал на документно-ориентированную систему управления базами данных MongoDB с открытым исходным кодом, не требующая описания схемы таблиц. Данная СУБД классифицирована как NoSQL, она использует JSON-подобные документы и схему базы данных. Скорость обеспечивается также тем, что MongoDB написана на языке C++. Запросы могут возвращать конкретные поля документов и пользовательские JavaScript-функции. Поддерживается поиск по регулярным выражениям.

MongoDB может работать с набором реплик, которые состоят из двух и более копий данных. Каждый экземпляр такого набора может в любой момент выступать в роли основной или вспомогательной реплики. Все операции записи и чтения по умолчанию осуществляются с основной репликой, а вспомогательные поддерживают в актуальном состоянии копии данных. В случае, когда основная реплика дает сбой, данный набор проводит выбор, который из реплик должен стать основным. Таким образом, второстепенные реплики могут дополнительно являться источником для операций чтения [2]. Все рассмотренные выше преимущества объясняют выбор MongoDB в качестве СУБД в разрабатываемом приложении.

Принцип работы и алгоритмы реализации. Система разделена на несколько подмодулей (рис. 1), каждый из которых отвечает за отдельный от остальных функционал.

В редактировании профиля можно изменять такие личные данные, как имя, фамилия, электронная почта, номер телефона, а также загружать фотографию, которая будет отображаться в левой панели сайта текущего пользователя, а также в его профиле (рис. 2). Для загрузки фотографии была использована библиотека «jQuery-Fileuploader», которая позволяет просто управлять процессом загрузки на JavaScript.

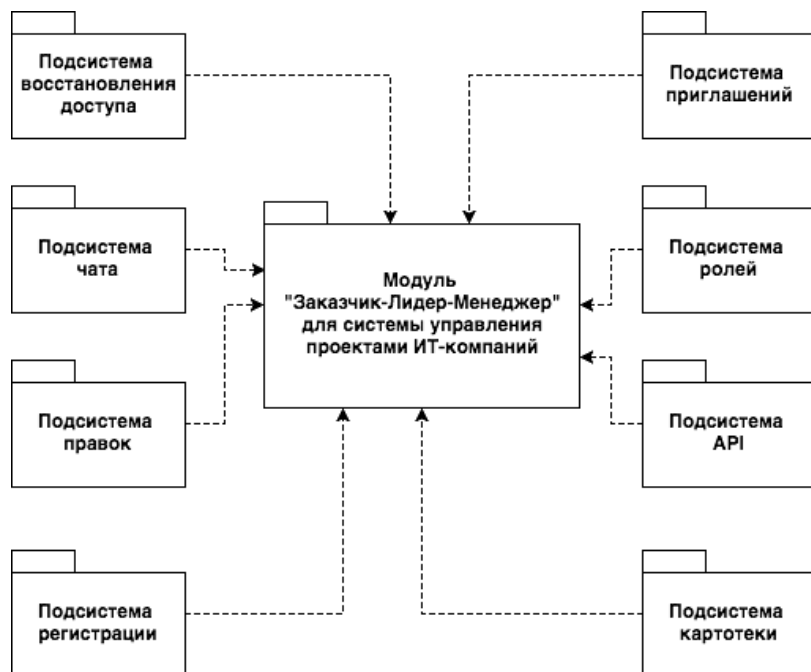


Рисунок 1. – Модули системы управления проектами

Рисунок. 2. – Редактирование профиля

Страница личных сообщений реализована с помощью технологии веб-сокетов. Она позволяет отправлять и принимать любые пакеты данных мгновенно без перезагрузки страницы. Для этого помимо клиентской части также требуется серверная реализация, которая позволит перенаправлять сообщения определенному пользователю в зависимости от полученной информации. Для реализации чата подобного типа была использована библиотека «SocketIO», которая включает в себя реализацию для клиентской

и серверной части веб-приложения. Если у адресата и адресанта открыты чаты друг с другом, то сервер понимает, что эти два пользователя общаются в данный момент. Для большей информативности еще до отправки, при наборе сообщений одним пользователем, другой пользователь видит, что его собеседник набирает сообщение. Это может быть полезно в том случае, если пользователь ждет моментального ответа, но не уверен, пользователь в сети или нет. Для того чтобы увидеть сообщения собеседника, необходимо кликнуть на его имя в левой панели чата. После нажатия будут загружены прошлые сообщения, а также непрочитанные. Все сообщения отсортированы по времени отправления так, что снизу выводятся последние (рис. 3). Слева выводятся сообщения собеседника, справа – сообщения текущего пользователя. По нажатию на фотографию собеседника, пользователь будет переадресован на личную страницу собеседника.

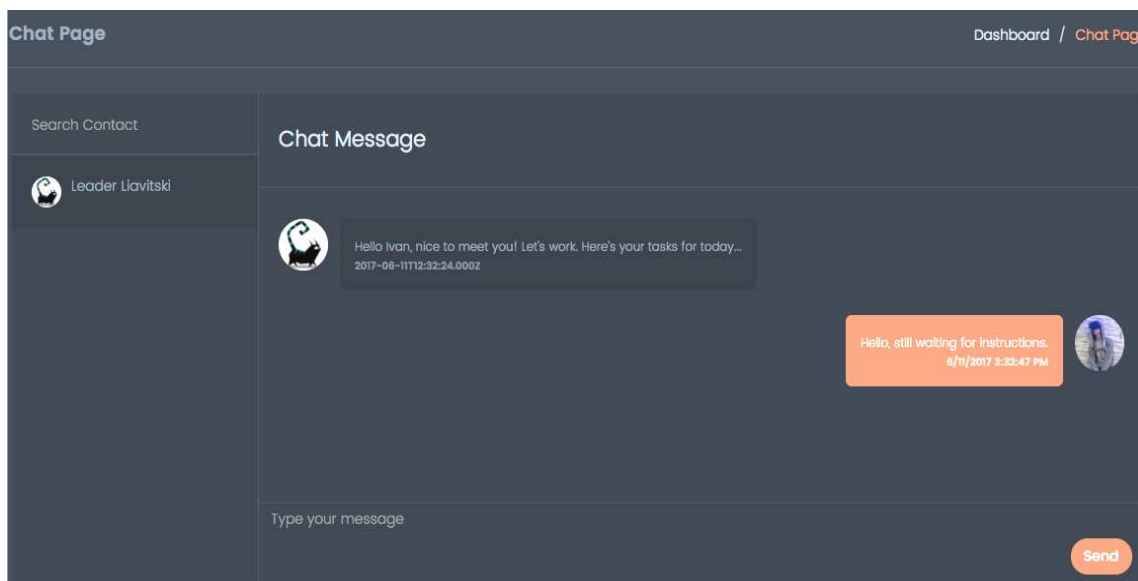


Рисунок 3. – Чат на WebSocket

В разделе «Invites» показаны текущие приглашения, которые еще не были активированы и по которым еще не зарегистрировались приглашенные пользователи. Если пользователь с введенным почтовым ящиком уже существует, то ему не будет отправлено приглашение повторно. При отправлении приглашения генерируется уникальная ссылка, при переходе на которую пользователь сможет зарегистрироваться. За один раз можно добавить неограниченное число сообщений, нажимая кнопку «Add more invites», при необходимости добавить очередное приглашение, при нажатии на которую добавится еще одна строка для ввода новых данных о пользователе. В таблице указать роль приглашаемого пользователя.

Заключение

Разработанная программная система с дружественным пользовательским интерфейсом как для исполнителей, так и для заказчиков позволит компаниям эффективно управлять заданиями заказчиков, эффективно вести переписку, назначать исполнителей, а также точно знать, какую именно задачу выполняет работник.

ЛИТЕРАТУРА

1. Node Package Manager [Электронный ресурс]. – Режим доступа: <https://npmjs.com>. – Дата доступа: 17.09.2017.
2. Базы данных [Электронный ресурс]. – Режим доступа: <http://www.bazydannyyh.ru/>. – Дата доступа: 18.09.2017.

УДК 004.4(004.773.3)

**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ПРИМЕНЯЕМОЕ ДЛЯ СБОРА
КОНФИДЕНЦИАЛЬНОЙ ИНФОРМАЦИИ СРЕДСТВАМИ ЭЛЕКТРОННОЙ ПОЧТЫ****А.В. ЛОБАНОВ***(Представлено: Е.С. ГАТИХО)*

Рассматриваются основные способы и механизмы отслеживания электронных сообщений. Приводятся примеры программного обеспечения, используемого для получения конфиденциальной информации средствами электронной почты.

В настоящее время почти каждый пользователь сети Интернет использует электронную почту для регистрации на различных сервисах общения, обмена корпоративной информацией и для прочих нужд. Сложно представить Интернет без электронных писем. Электронная почта – малозатратный и удобный способ связи в наше время. Однако она может быть применена для доступа к личным данным пользователя [1].

С каждым годом появляется все больше сервисов, направленных на отслеживание действий пользователей и на получение информации о получателях электронных писем.

Такие системы используют в своей работе следующую особенность электронной почты: содержании e-mail сообщения может представлять собой текст, а также может включать HTML. Это означает, что электронное письмо может содержать изображения, видео, файлы стилей и ссылки на различные ресурсы сети Интернет [2].

Принцип работы большинства сервисов отслеживания базируется на внедрении в письмо картинки, которая располагается на внешнем сервере. Когда получатель открывает письмо, идет запрос на почтовый сервис и в ответ приходит сплошной текст, который уже подгружается браузером или клиентским приложением пользователя, в этот момент скачиваются картинки, стили и прочие элементы HTML страницы с внешних серверов, один из которых может быть отслеживающим сервисом.

К основным функциям сервисов отслеживания электронных писем можно отнести:

- получение данных о местонахождении получателя во время открытия письма;
- определение точного времени открытия письма;
- получение данных об устройстве, с которого было прочитано электронное письмо;
- определение пользовательского окружения, такого как: операционная система и ее версия, используемое программное обеспечение для открытия.
- вывод статистики по пользователям, регионами областям [3].

Работа подобных сервисов схематично представлена на рисунке 1.

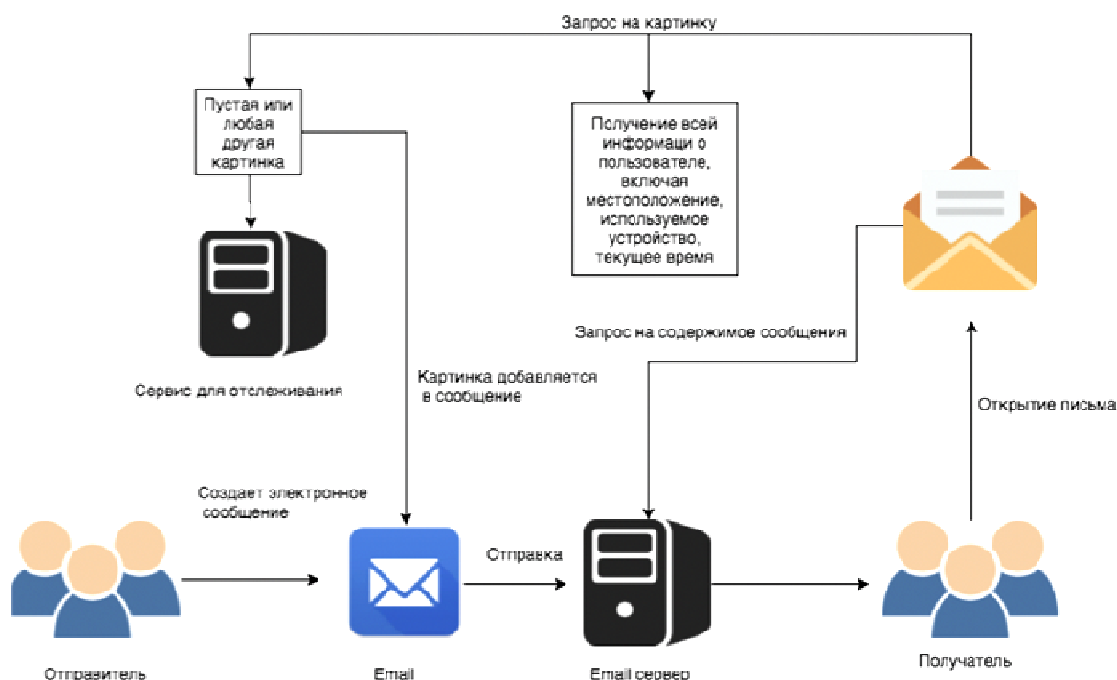


Рисунок 1. – Схема работы систем по отслеживанию сообщений

На текущий момент существует огромное количество сервисов для отслеживания электронных писем, все они предоставляют разный функционал – начиная от отслеживания открытия электронного письма и заканчивая сбором полной информации о пользователе и его действиях, во время открытия или прочтения электронного сообщения.

Список систем самых популярных систем отслеживания:

- Yesware – одна из самых востребованных систем отслеживания электронных писем, разработанная для бизнес-сегмента. Позволяет отслеживать информацию об открытии, кликах, загрузке презентаций и пр. Приложение признано одним из лучших для развития бизнеса по версии Forbes;
- ToutApp – аналог Yesware;
- Sidekick – система для увеличения продаж с помощью электронных сообщений;
- Mailtrack – популярная система отслеживания сообщений, общее количество скачиваний – около 1 млн., используется профессионалами из Yahoo;
- ContactMonkey – система интеграции систем продажи с Outlook-клиентом, использует систему отслеживания сообщений;
- Boomerang – бесплатная система отслеживания;
- CirrusInsight – популярная система отслеживания сообщений, используется такими компаниями, как LinkedIn, BBC;
- Streak – система отслеживания, используемая Твитером и Uber;
- Outreach – система отслеживания, используемая такими компаниями, как Microsoft, Adobe, Ebay [4].

Заключение

Системы отслеживания контента используются повсеместно, большинство из этих сервисов нужны для сбора маркетинговой информации и не затрагивают конфиденциальную информацию и направлены на улучшение для маркетинговых исследований. Но данный канал связи часто применяется для получения информации о пользователе, в целях отслеживания его местоположения и сбора другой конфиденциальной информации.

ЛИТЕРАТУРА

1. Электронная почта [Электронный ресурс]. – Режим доступа: [http://www.tadviser.ru/index.php/статья:электронная_почта_\(e-mail\)/](http://www.tadviser.ru/index.php/статья:электронная_почта_(e-mail)/). – Дата доступа: 20.09.2017.
2. EmailTracking [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Email_tracking. – Дата доступа: 24.09.2017.
3. Howtrackemail. [Электронный ресурс]. – Режим доступа: <https://www.hubspot.com/sales/email-tracking-guide>. – Дата доступа: 18.09.2017.
4. BestEmailTrackingServices [Электронный ресурс]. – Режим доступа: <https://adespresso.com/blog/5-best-email-tracking-services-know-when-your-email-has-been-opened/>. – Дата доступа: 25.09.2017.

УДК 004.056.53(004.773.3)

МЕТОДЫ ЗАЩИТЫ ОТ СРЕДСТВ СЛЕЖЕНИЯ ЗА ЭЛЕКТРОННОЙ ПОЧТОЙ**А.В. ЛОБАНОВ***(Представлено: Е.С. ГАТИХО)*

Рассматриваются основные способы и механизмы защиты от средств слежения за электронной почтой. Приводятся примеры технологий обеспечения конфиденциальности сокрытия реальной информации о пользователе во время работы с электронной почтой.

На данный момент остро встает вопрос о поиске средств, стратегий, принципов обеспечения Интернет безопасности и предотвращения несанкционированного получения какой-либо информации о конечном пользователе. Информационная безопасность является необходимым условием развития современного общества. Обеспечение информационной безопасности является одной из приоритетных задач многих государств. Сохранность конфиденциальности получаемой и передаваемой информации – это жизненно важный аспект в современном мире. Пользователю необходимо иметь средства защиты личных данных.

Существует несколько подходов по обеспечению защиты пользователя от почтовых сервисов слежения:

- отключение отображения изображений в электронном письме;
- ручное ограничение доступа для определенных адресов, используемых сервисами отслеживания [1].

Ни один из этих вариантов не является универсальным. При отключении изображений большая часть писем потеряет презентабельный вид, а также пользователь может вообще не увидеть необходимой важную информацию.

Во втором случае для каждого конкретного сервиса слежения необходимо собственноручно искать актуальные адреса, которые постоянно обновляются. Также стоит отметить, что количество сервисов постоянно растет. Поэтому этот вариант защиты представляется неудобным для обычного пользователя электронной почты [2].

Основная задача системы для защиты от слежения за электронной почтой состоит в сокрытии, подмены реальной информации о пользователе, чтобы сделать работу отслеживающих систем бессмысленной или невозможной. Самым логичным решением на первый взгляд является подмена изображения, но существует проблема при данном подходе – изменение содержимого сообщения возможно только в сторонних почтовых клиентах.

Система, которая бы могла полностью защищать пользователей во время работы с электронной почтой, должна удовлетворять следующим критериям:

- работа со всеми современными браузерами, так как большая часть пользователей использует только их при работе с электронной почтой;
- мобильные приложения для всех популярных платформ. По статистике, после браузеров, самым популярным методом для работы с электронной почтой являются мобильные устройства;
- расширения для популярных клиентов для работы с электронной почтой.

Если программное обеспечение удовлетворяет всем перечисленным качествам, она может гарантированно защитить пользователя от средств слежения, независимо от используемых платформ.

В случае с работой системы с расширениями для браузера или другими клиентами существует механизм защиты, основанный на перехватах загрузки изображений, и подмены отслеживаемого контента [3]. В момент загрузки всех изображений и другого контента он должен быть просканирован на соответствие определенным маркерам, которые, в свою очередь, указывают на то, что данный контент является отслеживаемым [4]. В этот момент возможно два варианта развития событий:

- изменение содержимого контента, либо полный отказ от его загрузки;
- передача контента другим пользователям в реальном времени и полная загрузка его на клиенте.

Первый способ обеспечивает высокий уровень защиты, но потенциально искажает конечное электронное сообщение. Второй же способ позволяет конечному пользователю скрыться среди множества других пользователей, которые загрузят контент по той же ссылке.

Основные *преимущества* применения разработанной системы:

- возможность прочтение электронной почты без видимых изменений структуры документа;
- обеспечение конфиденциальности данных.

Примерная схема защиты от систем отслеживания электронной почты изображена на рисунке 1.

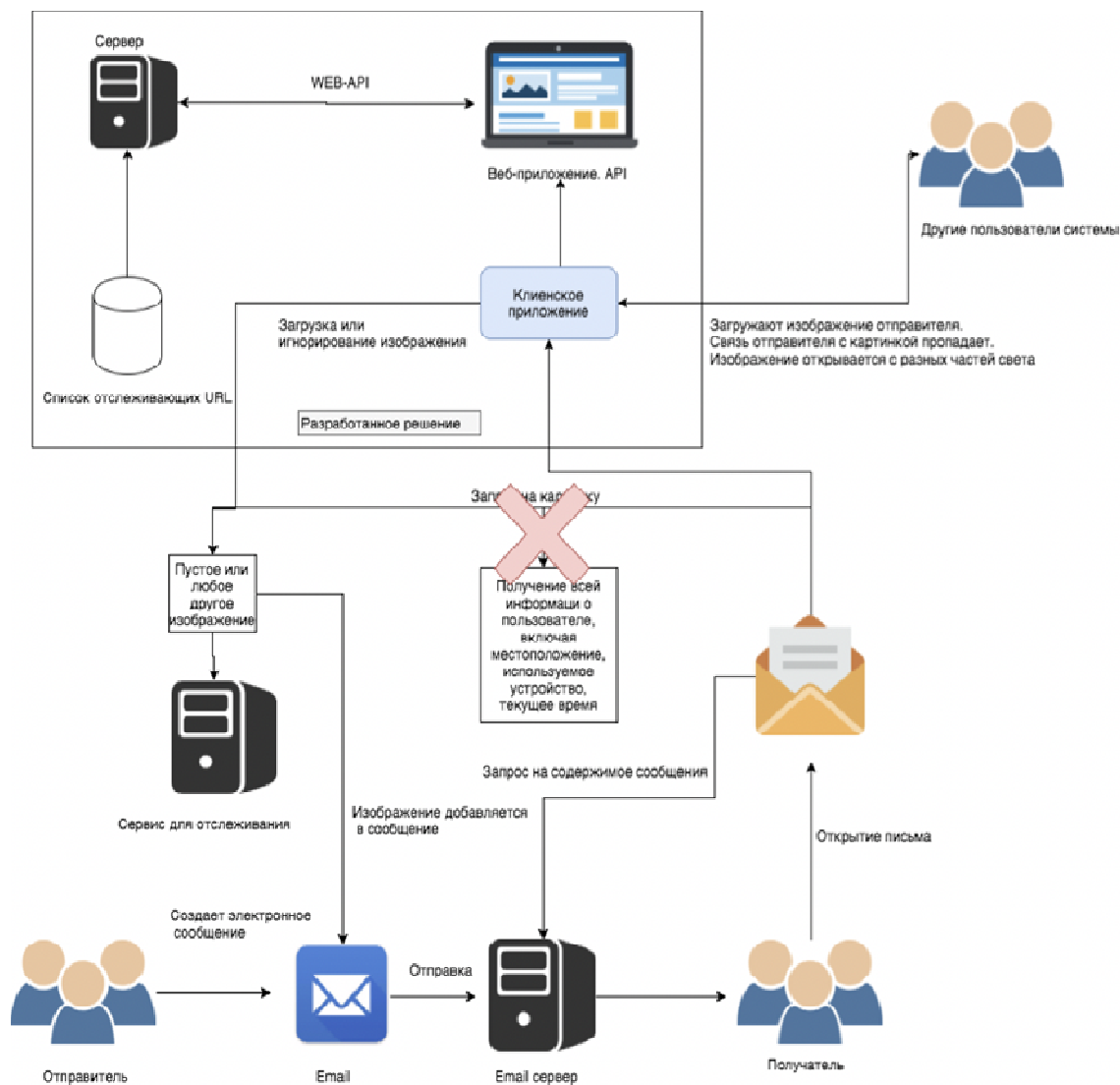


Рисунок 1. – Схема защиты от слежения за электронной почтой

ЛИТЕРАТУРА

1. Howstopemailtracking [Электронныйресурс]. – Режим доступа: <http://www.wikihow.com/Stop-Email-Tracking>. – Дата доступа: 20.09.2017.
2. Blockemailtracking [Электронныйресурс]. – Режим доступа: <https://nordvpn.com/blog/how-to-block-email-tracking/>. – Дата доступа: 24.09.2017.
3. Prevent loading images in browser [Электронныйресурс]. – Режим доступа: <http://www.thewindowsclub.com/disable-images-chrome-firefox-ie>. – Дата доступа: 18.09.2017.
4. Howtrackemailserviceswork [Электронныйресурс]. – Режим доступа: <http://blog.rocketbolt.com/how-does-email-open-tracking-work/>. – Дата доступа: 25.09.2017.

УДК 004.021

**МОБИЛЬНАЯ БАЗА ДАННЫХ REALM КАК АЛЬТЕРНАТИВА SQLITE И COREDATA
ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ IOS****Д.А. МЕДЮХО****(Представлено: Т.С. РУДЬКОВА)**

Анализируются теоретические сведения об особенностях мобильной базы данных Realm. Проведены синтетические тесты быстродействия между Realm, SQLite и CoreData с использованием итерационных и количественных методик.

Одной из главных проблем, с которой сталкиваются многие разработчики, является выбор базы данных. iOS-разработчик обладает небольшим выбором в этом вопросе: SQLite или CoreData. Однако существует еще одно незаурядное решение – мобильная база данных Realm. Она предоставляет разработчикам простой и понятный путь для хранения данных и запросов к ним.

Что такое Realm? Realm – кроссплатформенная мобильная база данных для iOS, доступна как версия для языка Swift, так и для Objective-C [1]. Realm создана для того, чтобы стать лучше SQLite и CoreData. Она не только лучше и быстрее, но и проста в использовании, так как вы можете сделать многое с помощью всего нескольких строк кода. Realm является совершенно бесплатной, и ее использование возможно без каких-либо ограничений. Realm создана для мобильных приложений и это новинка в сегменте мобильных баз данных, так как в последнее десятилетие инноваций в этой области замечено не было. Раньше у вас был только один вариант, которым является SQLite или любая другая обертка, которая использует SQLite.

Realm отличается простой в использовании, так как не является объектно-реляционной, а использует свои собственные наработки для достижения большей производительности и скорости выполнения.

Особенности Realm. Рассмотрим все преимущества и причины, почему использование Realm является лучшим выбором при работе с базой данных в мобильном приложении:

1. *Простая в установке.* Установить Realm легко – с помощью простой команды в Cocoapods, вы перейдете к работе с Realm.

2. *Быстрая.* Realm – невероятно быстрая библиотека для работы с базой данных. Realm быстрее, чем SQLite и CoreData, и сравнительные тесты здесь – лучшее доказательство для этого.

3. *Кроссплатформенная.* Файлы базы данных Realm кроссплатформенные и могут совместно использоваться iOS и Android.

4. *Масштабируемая.* Очень важно рассмотреть масштабируемость при разработке мобильного приложения, если приложение работает с большим количеством пользователей и большим количеством записей. Realm готова к масштабируемости и работе с большими объемами данных в кратчайшие сроки. Вы ускорите ваше приложение с помощью Realm.

5. *Хорошо документированная и есть отличная поддержка.* Команда Realm предоставила читаемую, хорошо организованную документацию о Realm. Если у вас возникли проблемы, вы можете связаться с ними через Twitter, Github или Stackoverflow.

6. *Надежная.* Realm используется большими компаниями в их мобильных приложениях, таких как Amazon, Google и Netflix [2].

7. *RealmBrowser.* Важный инструмент, который стоит внимания – это браузер Realm. Этот браузер помогает читать и редактировать Realm файлы. Эти файлы создаются в приложении со всей информацией о сущностях, атрибутах и записях в таблицах базы данных. Эти файлы могут использоваться на различных платформах, таких как iOS или Android.

8. *Бесплатная.* Все эти удивительные функции абсолютно бесплатны.

Сравнение быстродействия. Испытания проводились на реальной базе данных размером 150 000 записей. Realm оказалась в 2–3 раза быстрее SQLite и CoreData [3].

Синтетические тесты проводились с использованием iPhone SE на операционной системе iOS 11. Тестируемый набор данных – база данных, состоящая из одной таблицы и 150 000 записей.

Результаты прохода по всем записям запроса представлены на рисунке 1.

Результаты получения количества записей показаны на рисунке 2.

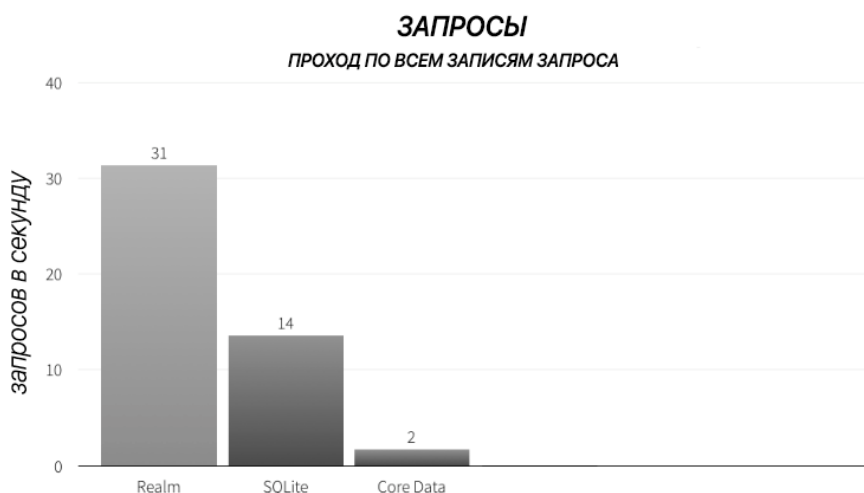


Рисунок 1. – Проход по всем записям запроса

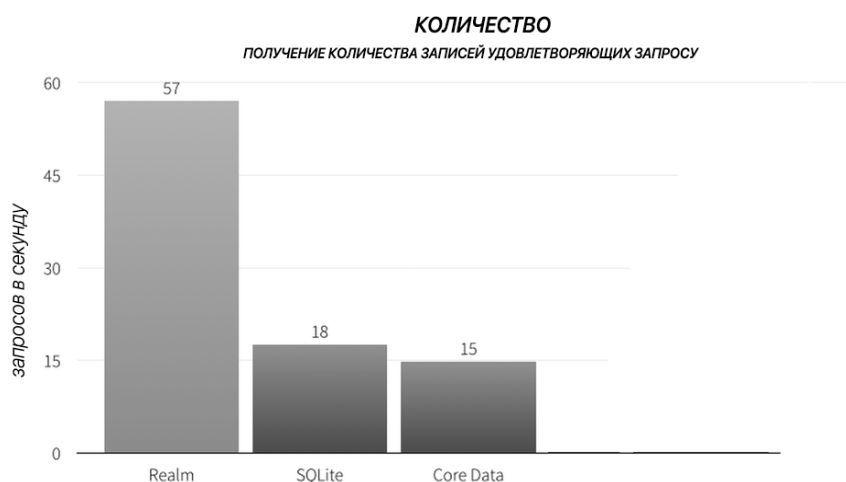


Рисунок 2. – Получение количества записей запроса

Как видно из итерационных и количественных методик тестирования, Realm показывает себя с лучшей стороны и оказывается быстрее своих конкурентов в 2–3 раза. В совокупности с простыми методами использования это дает мощный и одновременно легкий инструмент разработки [3].

Закключение

Realm является легким и простым решением для хранения данных. Realm обладает способностью масштабировать и упрощает работу всего за несколько строк кода, к тому же она быстрее SQLite и CoreData в 2–3 раза. Realm подойдет для большинства приложений и даже игр.

ЛИТЕРАТУРА

1. Realm // Wikipedia [Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/Realm>. – Дата доступа: 17.09.2017.
2. Realm [Электронный ресурс]. – Режим доступа: <https://realm.io/docs/objc/latest/>. – Дата доступа: 17.09.2017.
3. Realm Performance [Электронный ресурс]. – Режим доступа: <https://academy.realm.io/posts/realm-api-optimized-for-performance-and-low-memory-use>. – Дата доступа: 18.09.2017.

УДК 004.021

ШИФРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ДАННЫХ В БАЗЕ ДАННЫХ REALM ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ IOS

Д.А. МЕДЮХО

(Представлено: Т.С. РУДЬКОВА)

Анализируются теоретические сведения о методах защиты пользовательских данных в базе данных Realm. Рассмотрен практический подход использования шифрования и защищенного хранения ключей шифрования.

Каждую секунду пользователи со всего мира вводят личную информацию в установленные приложения, которые неявно обязуются хранить эту информацию в безопасности. Большое количество компаний и разработчиков по-прежнему относятся к безопасности, как аспекту, которым занимаются после реализации основных функций приложения, а не как к основному важнейшему компоненту их программного продукта.

Метод шифрования. В Realm имеется возможность шифрования файлов. Используя 64-байтовый ключ шифрования методом AES-256+SHA-2, Realm становится настолько безопасным, что многие банки начали использовать его в своих клиентских приложениях [1].

AES (Advanced Encryption Standard) – симметричный алгоритм блочного шифрования, принятый правительством США в качестве стандарта в результате конкурса, проведенного между технологическими институтами [2]. Он заменил устаревший DES (Data Encryption Standard), который больше не соответствовал требованиям сетевой безопасности, усложнившимся в XXI веке. Считается, что используемый в AES ключ длиной в 256 бит – достаточно надежная защита против лобовой атаки, то есть с чисто математической точки зрения подобрать один правильный пароль из всех возможных – трудно осуществляемая задача. Несмотря на некоторые недостатки AES, взломать защищенную с помощью этого алгоритма информацию практически нереально. Схема алгоритма показана на рисунке 1.

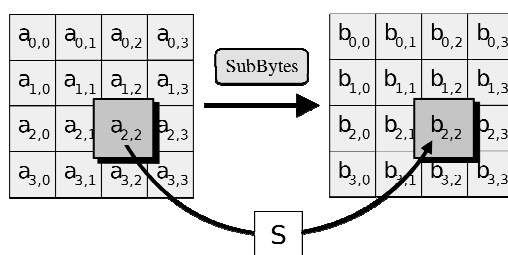


Рисунок 1. – Алгоритм AES Винсента Рэймена и Йона Даймена

Хэш-функции семейства SHA-2 пользуются высокой популярностью в приложениях, связанных с систематизацией, поиском и защитой информации [3]. Криптографические хэш-функции – это математические операции, выполняемые с цифровыми данными; сравнивая хеш (результат выполнения алгоритма) с известным значением хеша, человек может определить целостность данных. Ключевым аспектом криптографических хэш-функций является их устойчивость к взлому: никто не должен иметь возможность найти два разных входных значения, которые приводят к одному и тому же результату хэш-функции. SHA-2 справляется с этим на отлично и был выбран как стандарт хеширования данных. Схема алгоритма представлена на рисунке 2.

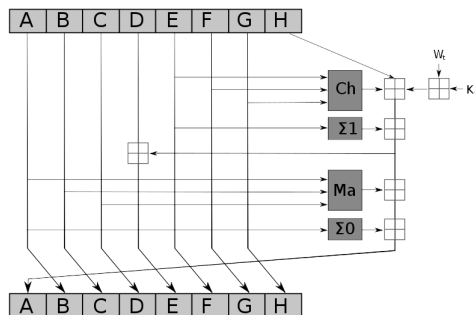


Рисунок 2. – Алгоритм SHA-2

Реализация. Включение шифрования в Realm настолько просто, что требуется только одна дополнительная настройка в настройках конфигурации Realm. Для того чтобы включить шифрование для файлов Realm, необходимо определить свойство *encryptionKey* при создании Configuration базы данных, а затем использовать созданную конфигурацию при создании объекта Realm [2]. Чтобы создать файл с шифрования, необходимо придерживаться следующих шагов:

- 1) генерация 64 байта случайных данных для ключа шифрования;
- 2) создание объекта конфигурации с ключом шифрования;
- 3) открытие файла Realm с созданной конфигурацией;
- 4) обработка исключений при открытии файла с неверным ключом;
- 5) использование объекта для дальнейших запросов.

После создания файла Realm с ключом шифрования, этот ключ будет необходим при каждом новом открытии. При попытке открыть файл с неверным ключом, Realm просто вернет ошибку, указав, что база данных недействительна.

Ограничения. Ключ шифрования для определенного файла Realm всегда только один и его изменение невозможно. Чтобы изменить ключ шифрования, можно экспортировать копию файла Realm, используя *writeCopyToPath(.:encryptionKey:)* и установить новый ключ на копию [2]. Также важно заметить, что шифрование в Realm увеличивает вашу базу данных на небольшой размер из-за необходимых операций шифрования и дешифрования.

Принцип защиты ключа шифрования. Хотя сами данные безопасно защищены Realm, самая важная проблема безопасности – защита ключа шифрования. Ответственность за хранение ключа шифрования несет разработчик. К этому вопросу должны относиться с тем же вниманием, что и к обработке пароля учетной записи пользователя. Самым безопасным вариантом хранения ключа шифрования является его сохранение Keychain. Keychain – надежно защищенный стандарт хранения паролей на устройствах пользователя [5].

Когда зашифрованный файл Realm скопирован на новое устройство, ключ шифрования также должен быть доступен на новом устройстве. Существует много способов, например, сохранить ключ шифрования в Keychain и синхронизировать Keychain через iCloud или через ваш собственный сервис при входе в систему [6].

Заключение

Процесс шифрования считается трудоемким и сложным, но с помощью Realm защита пользовательских данных становится простым шагом. Благодаря тому, что шифрование настолько легко интегрируется, это должно побудить разработчиков правильно защитить данные своих пользователей и свести к минимуму утечки данных в будущем.

ЛИТЕРАТУРА

1. Realm // Wikipedia [Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/Realm>. – Дата доступа: 17.09.2017.
2. Realm [Электронный ресурс]. – Режим доступа: <https://realm.io/docs/objc/latest/>. – Дата доступа: 17.09.2017.
3. Advanced Encryption Standard [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard. – Дата доступа: 17.09.2017.
4. SHA-2 [Электронный ресурс]. – Режим доступа <https://ru.wikipedia.org/wiki/SHA-2>. – Дата обращения 17.09.2017.
5. Adding Capabilities. iCloudKeychain Sharing [Электронный ресурс]. – Режим доступа: https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppDistributionGuide/Adding_Capabilities/Adding_Capabilities.html. – Дата доступа: 18.09.2017.
6. KeychainAccess [Электронный ресурс]. – Режим доступа: <https://support.apple.com/en-us/HT204085>. – Дата доступа: 18.09.2017.

УДК 004.223.2

**ПОДХОДЫ К РАЗРАБОТКЕ ИНТЕРФЕЙСА ДЛЯ ВЕБ-ПРИЛОЖЕНИЯ
ПО РАБОТЕ С МЕДИЦИНСКИМИ АМБУЛАТОРНЫМИ ДАННЫМИ ПАЦИЕНТОВ****Е.А. МЕНИЦКИЙ***(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)*

Представлен практический способ создания интерфейса для работы с медицинскими амбулаторными данными. Проведен анализ технологий, наиболее подходящих для разработки данной системы. Рассмотрена задача, реализация прототипа интерфейса на основе веб-технологий.

На сегодняшний день любое веб-приложение должно быть основано на общих поддерживаемых стандартах разработки, а также обладать понятным для пользователя интерфейсом. Разрабатываемое веб-приложение для работы с медицинскими амбулаторными данными пациентов позволяет производить работу с медицинскими данными, хранящимися в амбулаторных картах, а также удовлетворять требованиям к предоставляемым данным, по видам хранящейся информации и ее представлению.

В данном приложении процесс создания интерфейса приложения служит для прямого взаимодействия приложения с базой данных, сочетающей в себе всю необходимую информацию для вывода непосредственно в приложение. Интерфейс создается путем добавления необходимых компонентов и последующей их настройки. В интерфейсе любого приложения есть компоненты, обеспечивающие взаимодействие программы с пользователем.

Выбор технологий разработки интерфейса

Данное веб-приложение планируется разрабатывать с помощью языка программирования Java и фреймворка веб-разработки Spring Framework. Отсюда следует, что технологии разработки веб-интерфейса должны быть совместимы с данными технологиями, при этом предоставлять все возможности современных интерфейсов.

Разрабатываемое приложение по своей структуре и основной потребности должно давать возможности по работе и представлению большого количества данных, которые бы могли динамически меняться, при действиях пользователя. Анализ инструментов для создания веб-интерфейса позволил выявить несколько технологий, подходящих под данные цели:

JSP (Java Server Pages) – технология, позволяющая веб-разработчикам создавать содержимое, которое имеет как статические, так и динамические компоненты. Страница JSP содержит текст двух типов: статические исходные данные, которые могут быть оформлены в одном из текстовых форматов HTML, SVG, WML, или XML и JSP-элементы, которые конструируют динамическое содержимое. Кроме этого, могут использоваться библиотеки JSP-тегов, а также EL (Expression Language) для внедрения Java-кода в статичное содержимое JSP-страниц [1].

JSF (JavaServer Faces) – это фреймворк для веб-приложений, написанный на Java. Он служит для того, чтобы облегчать разработку пользовательских интерфейсов для Java EE-приложений. В отличие от прочих MVC-фреймворков, которые управляются запросами, подход JSF основывается на использовании компонентов. Состояние компонентов пользовательского интерфейса сохраняется, когда пользователь запрашивает новую страницу и затем восстанавливается, если запрос повторяется [2].

Обе технологии поддерживают возможность работы с динамическими данными, однако предпочтение было отдано технологии Java Server Pages, так как ее поддержка ведется наиболее интенсивно, а также данная технология имеет большую совместимость с Spring Framework благодаря работе через Servlet.

Также было решено добавить возможность адаптивности интерфейса для разрабатываемого приложения. Для этого использовали Bootstrap.

Bootstrap – свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографии, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая Java Script-расширения.

Bootstrap использует современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров [3].

Проектирование интерфейса

При разработке интерфейса, для удобства использования была применена вкладочная структура – информация логически разделена по вкладкам, что упрощает ее поиск, не загромождает главную страни-

цу и не приводит к необходимости создавать дополнительные формы, а также уменьшает время на загрузку других страниц (рис. 1).

Амбулаторная карта Пациенты

№ Карты	Участок	ФИО
2	30	Сергей Петров Сергеевич
3	45	Инна Сидорова Ивановна
5	45	Наталья Шланова Андреевна
6	3	Евгений Терпагин пилл
7	45	Иван Керчин Иванович
11	16	Александр Тетрович Сергеевич
12	16	Альмир Бажанов Сергеевич
13	1	Ольга Бажанова Сергеевна
14	15	Инга Воронова Андреевна
15	16	Серафима Жиглова Александровна
16	17	Сисей Жорж Иванович

Номер карты: 11
Участок: 16
ФИО: Александр Тетрович Сергеевич

Создать карту Редактировать Удалить

МИНИСТЕРСТВО ЗДРАВООХРАНЕНИЯ РБ

Форма по ОКУД
Учреждения по ОКПО
Медицинская документация
Форма ... 025/у

МЕДИЦИНСКАЯ КАРТА АМБУЛАТОРНОГО БОЛЬНОГО № 11

Фамилия, имя, отчество: Александр Тетрович Сергеевич
Пол: мужской Дата рождения: 1996-01-01 Телефон: 425678

Адрес больного: область: 16 населенный пункт: Могилев
район: Полоцкий улица (переулок): ул.Садовая
дом №: 16 корпус: 1 кв. №: 20

Место службы, работы: ОАО "МАЗ" отделение, цех: маркетинга

Профессия, должность: продавец

Лист диспансерного наблюдения Лист заключений Профилактические осмотры Лист вакцинации
Рентгелогические обследования Осмотр гинекологом Врачебные заключения Анамнез

Рисунок 1. – Прототип главной страницы для врача

Основная информация о пациентах хранится в таблицах, с которыми имеется возможность производить действия после выбора соответствующих данных. Такая структура является наиболее простой и понятной для пользователей с различным уровнем владения персональным компьютером. Использование стандартных элементов управления позволяет максимально снизить время обучения работе с приложением, а стилизация их с помощью шаблонов Bootstrap позволяет улучшить внешний вид и привлекательность данного приложения.

Также на странице были применены возможности загрузки и просмотра медицинских медиаданных. Осуществлялось это на тех же страницах, где происходит выбор данных пациентов. Данный подход реализован с помощью динамической погрузки данных JSP (рис. 2).

id	Дата	Тип	Доза(р/ч)	Результат	Подтверждение
0	2017-06-08	рентген стоп	10	перелом левой стопы	true
0	2017-06-23	рентген грудной клетки	10	легкие поля чистые	true

id: 0 Дата: 23.06.2017 Тип: рентген стоп Доза (р/ч): 10

Файл: Выбрать Подтверждение: ☒ Результат: перелом левой стопы

Добавить Редактировать Удалить Сохранить в XLS

L R

Рисунок 2. – Прототип страницы с медиа данными

Заключение

Правильным подходом при разработке веб-интерфейса является использование современных технологий, которые позволяют решать свои задачи во всех браузерах. Это, прежде всего, экономит время при разработке интерфейса, а также ресурсы на обработку данных, что является актуальным при большом количестве информации, которая подлежит постоянному изменению.

Рассмотренные технологии JSP и Bootstrap при взаимодействии позволяют создать адаптивный интерфейс любого уровня сложности, при этом сохраняя его функциональные возможности и привлекательный внешний вид. По итогам данной работы описана возможность их применения к построению интерфейса и динамической обработке запросов без перезагрузки страницы.

ЛИТЕРАТУРА

1. Java Server Pages // Википедия : свободная энцикл. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/JSP>. – Дата доступа: 25.09.2017.
2. Java Server_Faces // Википедии : свободная энцикл [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/JavaServer_Faces . – Дата доступа: 25.09.2017.
3. Bootstrap // Википедия : свободная энцикл. [Электронный ресурс]. – Режим доступа: [https://ru.wikipedia.org/wiki/Bootstrap_\(фреймворк\)](https://ru.wikipedia.org/wiki/Bootstrap_(фреймворк)). – Дата доступа: 25.09.2017.

УДК 004.223.2

**ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ
С МЕДИЦИНСКИМИ АМБУЛАТОРНЫМИ ДАННЫМИ ПАЦИЕНТА****Е.А. МЕНИЦКИЙ***(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)*

Рассматривается проектирование системы для работы с медицинскими амбулаторными данными, а также вопросы защиты этих данных. Проведен анализ технологий, наиболее подходящих для разработки данной системы. Проведены исследования по актуальности разработки данной системы.

На данный момент в Республике Беларусь, множество учреждений здравоохранения используют бумажный документооборот для хранения амбулаторных данных пациентов. Данный подход с момента развития информационных технологий значительно уступил в эффективности электронным медицинским системам хранения и обработки медицинских данных пациентов. Электронные системы имеют ряд преимуществ [1]:

- повышение качества управления медицинской, административной и финансовой деятельностью организации здравоохранения за счет комплексного внедрения электронной медицинской карты и электронного талона, предоставления дополнительных электронных услуг автоматического самообслуживания для пациентов;
- сокращение времени оформления медицинской документации, посредством получения готовых данных из баз данных и компьютерного ввода;
- сокращение сроков предоставления отчетов и реестров услуг на основе автоматически генерируемых шаблонов;
- оптимизация штата медицинского персонала учреждений здравоохранения;
- обеспечение оперативного контроля за показателями работы поликлиники, экспертизы качества оказания медицинской помощи, посредством логирования данных и составления отчетной выборки;
- повышение достоверности информации о показателях лечебно-диагностической работы и финансовой деятельности посредством методов криптографической защиты информации;
- повышение пропускной способности учреждений здравоохранения за счет сокращения времени работы с амбулаторной картой и другими документами, и как следствие – сокращение очередей;
- предоставление возможностей телемедицины, для коммуникации по врачам и пациентам, а также консультации врача с врачом.

Данные критерии, несомненно, доказывают эффективность электронных систем, а также актуальность их разработки и внедрения в систему здравоохранения.

Средства решения задачи

Основу разрабатываемой системы составляет возможность хранить, накапливать и обрабатывать данные. Медицинские данные должны храниться в строго систематизированном виде, основываясь на нормах ведения медицинской документации, и иметь явную взаимосвязь.

Основываясь на этих данных, необходимо будет использовать реляционную базу данных для хранения сущностей и язык запросов SQL для построения базы данных. В данном случае использование именно реляционной базы данных позволит формально и однозначно определить ограничения целостности БД в терминах ее концептуальной схемы.

Так как медицинские данные строго регламентированы, при разработке системы целесообразным будет использовать официальные документы по составлению амбулаторных медицинских карт [2] и представлять отдельный лист вида обследования или сбора данных, как отдельную сущность, связанную с пациентом.

Очевидно, что при реализации приложения важно сформировать некоторую архитектуру. Для взаимодействия с базой данных подойдет паттерн MVC, представляющий собой взаимодействие 3-х элементов: модель, представление и контроллер [3]. Модель является некоторыми данными, которые описывают предметную область, и над ними могут быть выполнены некоторые операции. Также эта информация может быть предоставлена пользователю на графическом интерфейсе.

Представление и является этим самым графическим интерфейсом. Через данный интерфейс пользователи системы будут передавать данные на выполнение задачи контроллером. Контроллер, в свою очередь, реализует некоторые алгоритмы для получения данных с представления, обработки их и отправки обратно.

Также одним из преимуществ подобных систем является возможность передачи информации в виде потокового видео-сообщения. Реализация данного функционала позволит проводить видеокон-

сультации и осмотры, что, несомненно, уменьшит время на обслуживание пациента, тем самым позволит разгрузить учреждения здравоохранения. Для реализации данной системы, нам подойдет технология WebRTC.

Технология WebRTC – это технология, позволяющая интернет-приложениям производить захват видео и аудио- и медиапоток опционально, не прибегая к посредническим сервисам и протоколам. Набор стандартов, которые включает в себя технология WebRTC, позволяет обмениваться данными и проводить телеконференции в режиме узел-узел [4].

Общая архитектурная схема веб-приложения для работы с медицинскими амбулаторными данными пациента представлена на рисунке 1.

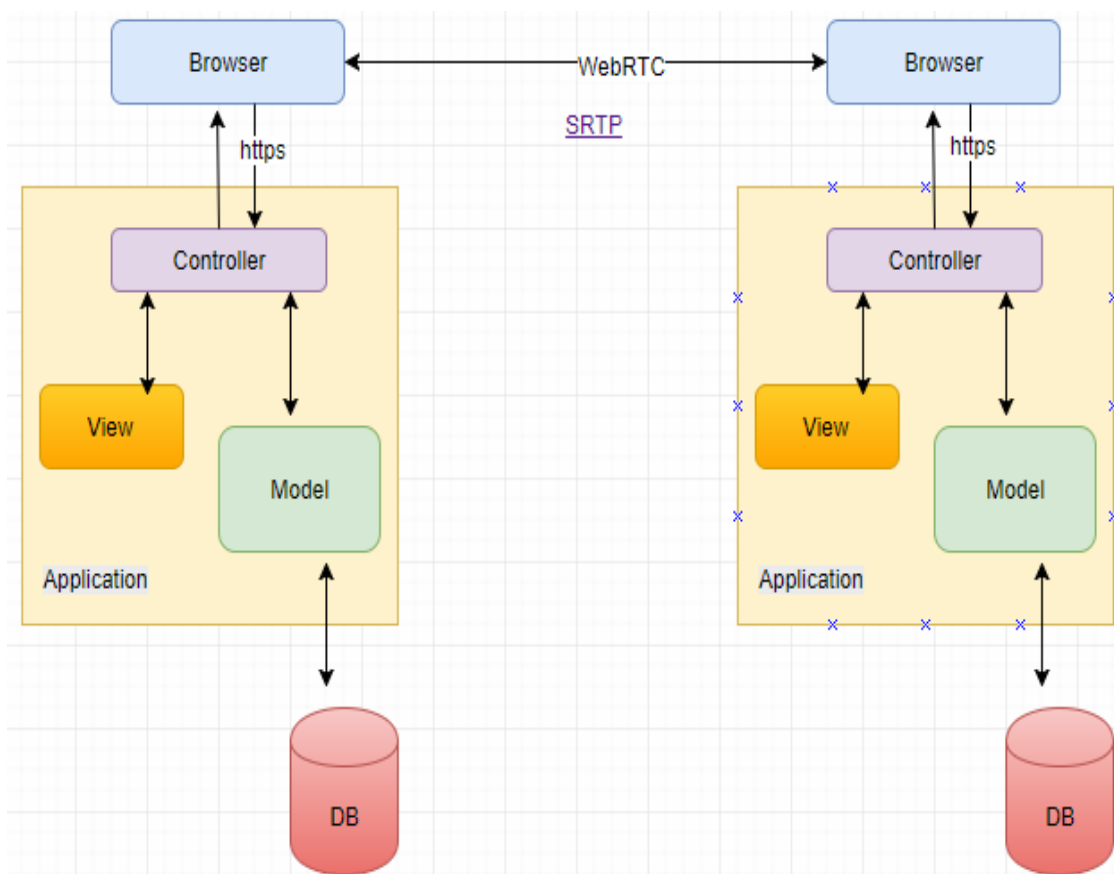


Рисунок 1. – Общая архитектурная схема веб-приложения для работы с медицинскими амбулаторными данными пациента

Принцип защиты информации

Основываясь на том, что данная система предназначена для хранения конфиденциальных данных пациентов, необходимо разработать систему аутентификации, и криптографической защиты данных.

Для предоставления аутентификации, необходимо создать сущность пользователя в базе данных с различными видами ролей, такими как пациент, врач, администратор и т.п., при этом разделить уровни доступа к алгоритмам работы с данными, согласно реальному уровню доступа к медицинской документации. Также разделения прав на использование алгоритмов должно быть реализовано как на уровне базы данных, через редактор ролей, так и через контроллер, с помощью проверок на этапе кодирования.

Для защищенной передачи данных по сети интернет предназначены следующие виды технологии:

- *протокол HTTPS* – стандарт передачи данных между различными машинами, который определяет, что должно выступать в качестве сигнала начала передачи, как обозначаются данные и т.д. При этом данные шифруются по протоколу SSL, что делает проблематичным не только перехват, но и получение конфиденциальной информации [5];

- *протокол SRTP* – это профиль расширения RTP (Real-Time Transport Protocol, транспортный протокол в реальном времени), который добавляет дополнительные функции безопасности, такие как аутентификация сообщений, конфиденциальность и защита от прослушивания. Данный протокол будет использован при передаче видеосообщений [6];

- *шифр AES* – симметричный алгоритм блочного шифрования. Данный вид шифрования будет использовать один и тот же симметричный ключ. Для разрабатываемой системы это является оптимальным вариантом, так как подобные системы также защищены юридическими соглашениями о неразглашении третьим лицам. Используемый метод шифрования будет использован для шифрации конфиденциальных данных на уровне базы данных.

Заключение

В ходе данного исследования спроектирована программная система, наиболее подходящая по архитектурным особенностям под основу для разработки веб-приложения для работы с медицинскими амбулаторными данными пациента. При этом разработанная архитектура является универсальной для любого современного языка программирования, а также оставляет возможности для реализации необходимого функционала, обеспечивая при этом высокий уровень защиты данных.

ЛИТЕРАТУРА

1. Информатизация амбулаторно-поликлинического звена [Электронный ресурс]. – Режим доступа: http://minzdrav.gov.by/ru/static/informatization/inform_apzveno. – Дата доступа: 23.09.2017.
2. Оформление медицинской карты амбулаторного больного [Электронный ресурс]. – Режим доступа: https://www.bsmu.by/downloads/kafedri/k_poli_ter/stud/v1.pdf. – Дата доступа: 23.09.2017.
3. Model-View-Controller [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Model-View-Controller>. – Дата доступа: 23.09.2017.
4. WebRTC API [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/API/WebRTC_API. – Дата доступа: 23.09.2017.
5. Протокол HTTPS – что такое? [Электронный ресурс]. – Режим доступа: <http://fb.ru/article/221368/protokol-https---chto-takoe>. – Дата доступа: 23.09.2017.
6. SRTP – что такое безопасный протокол передачи данных в реальном времени? [Электронный ресурс]. – Режим доступа: <https://www.3cx.ru/webrtc/srtp/>. – Дата доступа: 23.09.2017.

УДК 004.853

БИБЛИОТЕКА ГЛУБИННОГО ОБУЧЕНИЯ DEEPLARNING4J

А.А. СОЛОВЬЁВ

(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)

Анализируется библиотека с открытым исходным кодом, распределенная библиотека глубинного обучения для виртуальной машины Java (JVM) Deeplearning4J (DL4J) на примере архитектуры сверточной нейронной сети, представленной Алексом Крижевским (AlexKrizhevsky) на конкурсе ImageNet в 2012 году.

Стремительное развитие технологий машинного обучения делает спрос на различные инструменты, позволяющие обучать архитектуры нейронных сетей любой сложности, а также тонко настраивать различные параметры при обучении. Одним из современных инструментов машинного обучения является библиотека DL4J.

DL4J – это первая коммерческая библиотека глубинного обучения с открытым исходным кодом, написанная для Java и Scala [1]. Данная библиотека позволяет строить архитектуры нейронных сетей различной сложности и глубины, тонко настраивать всевозможные параметры обучения. Библиотека предназначена для использования в бизнес-средах на распределенных графических и центральных процессорах, включает в себя реализацию ограниченной машины Больцмана, глубокой сети убеждений, глубокого автокодера, многоуровневого автокодирования и рекурсивной сети нейронных тензоров, word2vec, doc2vec и GloVe. Все эти алгоритмы включают распределенные параллельные версии, которые интегрируются с ApacheHadoop и Spark [2].

Для анализа данной библиотеки рассмотрим архитектуру сверточной нейронной сети AlexNet, которая заняла первое место в конкурсе ImageNet в 2012 году [3].

Программная реализация данной архитектуры сверточной нейронной сети при помощи библиотеки DL4J представлена следующим образом:

```
01: MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
02: .seed(seed)
03: .weightInit(WeightInit.DISTRIBUTION)
04: .dist(new NormalDistribution(dist))
05: .activation(Activation.RELU)
06: .updater(Updater.NESTEROVS).momentum(momentum)
07: .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
08: .iterations(iterations)
09: .gradientNormalization(GradientNormalization.RenormalizeL2PerLayer)
10: .learningRate(learningRate)
11: .regularization(true).l2(l2)
12: .list()
13: .layer(0, new ConvolutionLayer.Builder(kernelSize, stride, padding)
    .nIn(in).nOut(out).build())
14: .layer(1, new LocalResponseNormalization.Builder().build())
15: .layer(2, new SubsamplingLayer.Builder(kernelSize, stride).build())
16: .layer(3, new ConvolutionLayer.Builder(kernelSize, stride, padding)
    .nIn(in).nOut(out).build())
17: .layer(4, new LocalResponseNormalization.Builder().build())
18: .layer(5, new SubsamplingLayer.Builder(kernelSize, stride).build())
19: .layer(6, new ConvolutionLayer.Builder(kernelSize, stride, padding)
    .nIn(in).nOut(out).build())
20: .layer(7, new ConvolutionLayer.Builder(kernelSize, stride, padding)
    .nIn(in).nOut(out).build())
21: .layer(8, new ConvolutionLayer.Builder(kernelSize, stride, padding)
    .nIn(in).nOut(out).build())
22: .layer(9, new SubsamplingLayer.Builder(kernelSize, stride).build())
23: .layer(10, new DenseLayer.Builder().nOut(out).dropOut(dropOut).build())
```

```
24: .layer(11, new DenseLayer.Builder().nOut(out).dropOut(dropOut).build())
25: .layer(12, new OutputLayer.Builder(LossFunctions.NEGATIVELOGLIKELIHOOD)
    .nOut(out).activation(Activation.SOFTMAX).build())
26: .backprop(true)
27: .setInputType(InputType.convolutional(height, width, channels))
28: .build().
```

Рассмотрим данный программный код подробнее. В первой строке программного кода происходит объявление и инициализация многослойной нейронной сети. Создается объект класса `MultiLayerConfiguration`, которому присваивается значение нового объекта класса `NeuralNetConfiguration`. Данный класс реализует методы конфигурирования нейронных сетей, которые охватывают параметры каждого слоя.

Далее устанавливаются необходимые нам параметры нейронной сети. Метод `seed` принимает в себя вещественное число, которое используется при инициализации весов нейронной сети. Инициализация весов происходит по определенному закону, который устанавливается в методе `weightInit`. Тип закона устанавливается в методе `dist`. В нашем случае инициализация весов происходит по нормально-распределенному закону.

Также необходимо указать функцию активации для скрытых слоев нейронов. Для этого используется метод `activation`, который в качестве параметра принимает вид функции. В настоящее время огромной популярностью пользуется выпрямленная линейная функция активации `RELU`. Использование данной функции приводит к прореживанию весов, что ограждает нейронную сеть от переобучения.

После в методе `updater` указывается механизм обновления весов. В большинстве случаев хорошим выбором является использование алгоритма оптимизации стохастического градиентного спуска `NESTEROVS` [1]. Метод `momentum` является дополнительным методом, который отвечает за то, насколько быстро алгоритм оптимизации сходится в оптимальной точке.

Метод `iterations` принимает целое число, которое указывает на количество итераций для одной эпохи обучения сети.

Градиентная нормализация (`gradientNormalization`) применяется во избежание проблем малых и больших градиентов.

В методе `learningRate` указывается скорость обучения нейронной сети. Данный параметр является одним из важнейших при обучении сети. Высокая скорость обучения сети означает, что в обновлениях веса будут делаться крупные шаги, поэтому образцу может потребоваться меньше времени, чтобы набрать оптимальный набор весов. Но слишком высокая скорость обучения может привести к крупным и недостаточно точным скачкам, которые помешают достижению оптимальных показателей.

В следующем методе мы указываем нашей сети, что будем использовать регуляризацию.

Регуляризация помогает избежать переобучения нейронной сети. В нашем случае мы используем `L2`-регуляризацию. Регуляризация `L2` накапливает большие весовые коэффициенты сети и предотвращает слишком большой вес. Общие значения для `L2`-регуляризации составляют от $1e-3$ до $1e-6$.

Функция `list` позволяет создать многослойную нейронную сеть. В архитектуре рассматриваемой сети находится: 13 слоев, из которых 5 сверточных слоев; 3 слоя субдискретизации; 2 нормализации; 2 полносвязных слоя и выходной слой.

Метод `layer` принимает два параметра – номер слоя и объект класса `Layer`. Рассматриваемые далее классы слоев наследуются от класса `Layer`.

Класс `ConvolutionLayer` реализует метод `builder`, в котором устанавливаются такие параметры, как размер ядра свертки, шаг свертки и отступ. Также класс реализует методы `nIn` и `nOut`, в которых устанавливаются количество входных и выходных нейронов; метод `build` – конфигурирует заданные параметры.

Класс `SubsamplingLayer` также реализует метод `builder`, в котором указываются размер ядра и шаг субдискретизации. Данный класс поддерживает следующие типы субдискретизации: `MAX`, `AVG`, `NON`. Слои данного типа предназначены для уменьшения размерности карт признаков, полученных после свертки. Уменьшение размерности карт признаков приводит к сокращению параметров, используемых при обучении, без существенной потери важной информации.

Класс `LocalResponseNormalization` реализует алгоритм нормализации данных.

Класс `DenseLayer` реализует функционал полносвязного слоя прямого распространения. Также в данном классе реализован метод `dropOut`, который принимает вещественное значение, указывающее на процент нейронов, которые будут отброшены во время обучения [4]. Данный параметр позволяет сети не переобучаться.

Класс `OutputLayer` реализует функционал вывода данных с различными объективными совпадениями для разных целей. Данный класс производит классификацию, а также регрессию. Метод `Builder`

данного класса принимает в качестве аргумента некоторую функцию. В нашем случае это функция потерь отрицательного логарифмического правдоподобия.

Метод `backprop` реализует механизм обратного распространения ошибки, в качестве параметра принимает значение типа `bool`, которое указывает, использовать ли данный метод при обучении сети.

В методе `setInputType` устанавливается тип нейронной сети (в нашем случае сверточная), а также размер и количество каналов входных изображений.

Метод `build` собирает описанную выше нейронную сеть.

Заключение

В данной работе были рассмотрены лишь некоторые возможности библиотеки DL4J и проанализированы основные моменты при построении сверточной нейронной сети. Достоинствами данной библиотеки являются хорошая документация классов, методов и алгоритмов, огромное количество всевозможных методов обучения нейронных сетей, а также то, что данная библиотека написана для виртуальной машины Java. В библиотеке содержится большое количество методов конфигурирования нейронных сетей, однако основополагающими методами являются методы инициализации весов (`weightInit`), установки функции активации (`activation`) и скорости обучения сети (`learningRate`), а также конфигурирования слоев (`layer`) и способа их обучения. Также стоит отметить, что DL4J – это библиотека с открытым исходным кодом, что позволяет свободно использовать и распространять исходные коды с использованием данной библиотеки.

ЛИТЕРАТУРА

1. Библиотека глубинного обучения DeepLearning4J [Электронный ресурс]. – Режим доступа: <https://deeplearning4j.org/>. – Дата доступа: 23.09.2017.
2. Библиотека с открытым исходным кодом DeepLearning4J [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Deeplearning4j/>. – Дата доступа: 24.09.2017.
3. Krizhevsky, A. Imagenet classification with deep convolutional neural networks / Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton // NIPS. – 2012. – 1106–1114 p.
4. Dropout: A Simple Way to Prevent Neural Networks from Overfitting / Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever // Journal of Machine Learning Research. – 2014. – (15). – P. 1929–1958.

УДК 004.021

**ОБНАРУЖЕНИЕ ДЫМА ОТ ПОЖАРА С ПРИМЕНЕНИЕМ ОБРАБОТКИ
НА ОСНОВЕ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ****А.А. СОЛОВЬЁВ***(Представлено: канд. физ.-мат. наук, доц. Д.Ф. ПАСТУХОВ)*

Описывается алгоритм детектирования дыма от пожара, реализованный на языке программирования Java. Приведены примеры работы разработанного приложения и сделаны выводы о целесообразности использования сверточных нейронных сетей для решения поставленной задачи.

В связи с интенсивным развитием вычислительной техники, системы машинного зрения стали доступны широкому классу пользователей и позволяют эффективно решать многие задачи в практической деятельности человека [1; 2]. Одной из актуальных прикладных задач, решаемых в последнее время с помощью систем машинного зрения, является обнаружение лесных пожаров. Для решения данной задачи был разработан алгоритм обработки динамических изображений с применением глубокого обучения, в частности сверточных нейронных сетей (СНС). Разработанный алгоритм включает в себя следующие три стадии выполнения программы: цветовая сегментация, обнаружение движущихся областей и классификация выделенных предполагаемых дымчатых областей.

Локализация объектов в сцене наблюдения происходит в два основных этапа. Сначала в видеопоследовательности осуществляется поиск движущихся объектов, после чего происходит их классификация по цветовым характеристикам.

В качестве метода обнаружения движения был выбран метод вычитания фона, основанный на смеси Гауссовых распределений. Данный метод позволяет обнаруживать медленно движущиеся объекты в сцене.

Классификацию движущихся объектов было решено проводить в цветовом пространстве HSV, так как классификация в данном цветовом пространстве дает меньше ложных обнаружений дыма, чем в цветовом пространстве YUV. Классификатором локализованных областей является обученная сверточная нейронная сеть. В качестве архитектуры нейронной сети выбрана модель LeNet [3], которая является простой, не требующей больших вычислительных ресурсов и позволяющей производить обучение.

Общая схема разработанного алгоритма представлена на рисунке 1.

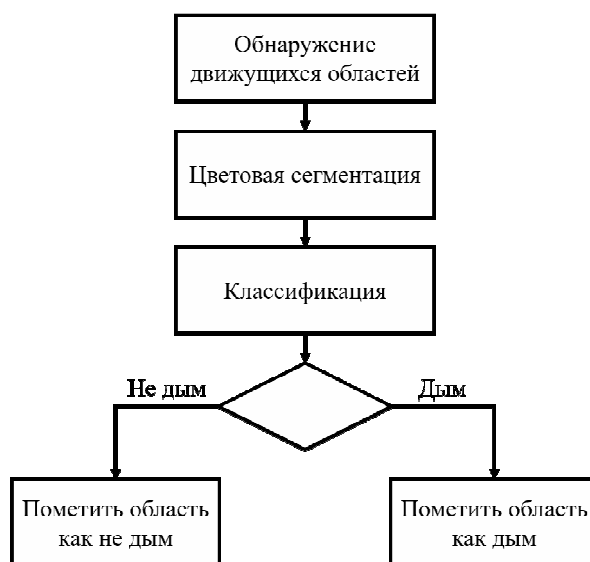


Рисунок 1. – Общая схема разработанного алгоритма

Для проведения экспериментов и оценки результатов исследований разработанный алгоритм был реализован на языке Java с использованием интегрированной среды разработки программного обеспечения IntelliJIDEA, библиотеки компьютерного зрения OpenCV 3.0 и библиотеки глубокого обучения DL4J. При проектировании графического интерфейса было принято решение использовать платформу для создания приложений JavaFX.

Разработка, тестирование и анализ результатов работы алгоритма проводились с помощью ноутбука Samsung 300E5Z со следующими основными характеристиками: центральный процессор – IntelCore i3-2350M с тактовой частотой 2,3 ГГц; объем ОЗУ – 4,0 Гб; графический ускоритель – IntelHDGraphics 3000.

Для захвата видеопотока (с камеры видеонаблюдения либо из постоянной памяти ЭВМ) и извлечения кадров из видеопоследовательности использовалась библиотека OpenCV. OpenCV (OpenSource-ComputerVisionLibrary) – библиотека алгоритмов компьютерного зрения, обработки изображений с открытым исходным кодом [4]. OpenCV создана для обеспечения общего стандартного интерфейса машинного зрения для приложений в области обработки динамических изображений и увеличения использования систем машинного зрения в коммерческих продуктах.

Построение, обучение и использование СНС проводилось при помощи библиотеки DL4J (DeepLearningforJava) – библиотека глубинного обучения, написанная для Java и виртуальной машины Java (JVM), и вычислительная среда с широкой поддержкой алгоритмов глубокого обучения [5].

Для демонстрации возможностей алгоритма было решено реализовать приложение, предназначенное для раннего детектирования лесных пожаров. Интерфейс реализованного приложения представлен на рисунке 2. Интерфейс приложения состоит из главного окна, окна вывода текстовой информации и окна вывода видеoinформации.

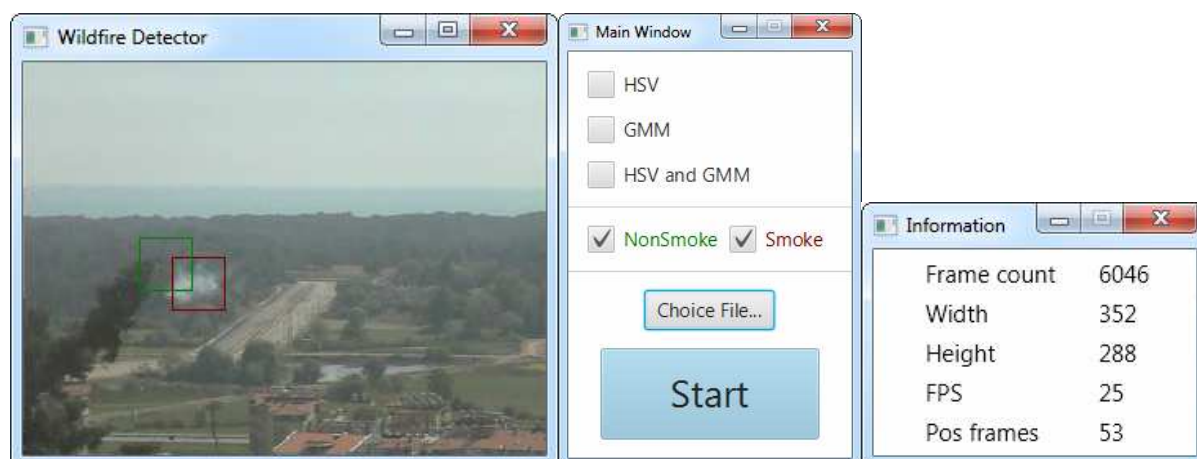


Рисунок 2. – Интерфейс программного продукта

Для анализа разработанного алгоритма использовались видеопоследовательности, полученные в условиях реального освещения при съемке вне помещения с различным углом наклона, в том числе и с нестабильным фоном. Работа разработанного приложения представлена на рисунке 3.



Рисунок 3. – Кадры используемых видеопоследовательностей

Заключение

Результаты тестирования разработанного алгоритма показали, что сверточные нейронные сети могут служить в качестве классификатора пожара. Ввиду того, что в качестве классификатора была выбрана простая модель сети, а при обучении использовалась относительно небольшая выборка примеров, тестирование показало плохие результаты. Коэффициент ложноположительной классификации разработанного алгоритма оказался равен 7%, что недопустимо. Однако можно предположить, что при применении сложных архитектур нейронных сетей и большой обучающей выборки результаты детектирования пожара могут быть улучшены многократно.

ЛИТЕРАТУРА

1. Методы автоматического обнаружения и сопровождения объектов. Обработка изображений и управление / Б.А. Алпатов [и др.] – М. : Радиотехника, 2008. – 176 с.
2. Ngan, K. Video Segmentation and Its Applications / K. Ngan, H. Li // Springer, 2011. – 165 p.
3. LeCun Yann. Gradient-based learning applied to document recognition / Yoshua Bengio, Yann LeCun, Leon Bottou, Patrick Haffner // IEEE, 1998.
4. Библиотека компьютерного зрения OpenCV [Электронный ресурс]. – Режим доступа: <http://opencv.org/>. – Дата доступа: 27.09.2017.
5. Библиотека глубинного обучения DL4J [Электронный ресурс]. – Режим доступа: <https://deeplearning4j.org/>. – Дата доступа: 27.09.2017.

УДК 004.8

СПОСОБЫ МАШИННОГО ОБУЧЕНИЯ

М.А. ШЕВЦОВ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматриваются способы машинного обучения, отдельные из них поддерживают несколько видов обучения, некоторые могут быть реализованы только с определенным алгоритмом обучения. В данной работе рассмотрены основные из рассмотренных способов.

1. Обучение с учителем (англ. **supervised learning**) – один из способов машинного обучения, в ходе которого испытуемая система принудительно обучается с помощью примеров «стимул-реакция».

Между входами и эталонными выходами (стимул-реакция) может существовать некоторая зависимость, но она неизвестна. Известна только конечная совокупность пар «стимул – реакция», называемая обучающей выборкой. На основе этих данных требуется восстановить зависимость (построить модель отношений стимул – реакция, пригодных для прогнозирования), то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ.

Под учителем понимается либо сама обучающая выборка, либо тот, кто указал на заданных объектах правильные ответы.

Постановка данного эксперимента предполагает наличие экспериментальной системы, метода обучения и метода испытания системы или измерения характеристик.

Основные алгоритмы:

- линейная и логистическая регрессия;
- машина опорных векторов;
- наивный Байес;
- нейронная сеть;
- повышение градиента;
- классификационные деревья и случайный лес.

2. Обучение без учителя (англ. **unsupervised learning**) – способ машинного обучения, при котором испытуемая система спонтанно обучается выполнять поставленную задачу без вмешательства со стороны экспериментатора. Метод пригоден для задач, в которых известны описания множества объектов (обучающей выборки) и требуется обнаружить внутренние взаимосвязи, зависимости, закономерности, существующие между объектами.

Обучение без учителя часто противопоставляется обучению с учителем, когда для каждого обучающего объекта принудительно задается «правильный ответ», и требуется найти зависимость между стимулами и реакциями системы.

Экспериментальная схема обучения без учителя часто используется в теории распознавания образов, машинном обучении. При этом в зависимости от подхода формализуется в ту или иную математическую концепцию.

Основные алгоритмы:

- алгоритм кластеризации, такой как K-mean, иерархическая кластеризация или модели смеси;
- алгоритмы сокращения размерности, такие как PCA, ICA или autoencoder;
- обнаружения аномалий, чтобы найти выбросы в данных, т.е. наблюдения, которые не соответствуют шаблонам набора данных.

3. Обучение с подкреплением (англ. **reinforcement learning**) – один из способов машинного обучения, в ходе которого испытуемая система (агент) обучается, взаимодействуя с некоторой средой. Откликом среды (а не специальной системы управления подкреплением, как это происходит в обучении с учителем) на принятые решения являются сигналы подкрепления, поэтому такое обучение является частным случаем обучения с учителем, но учителем является среда или ее модель. Также нужно иметь в виду, что некоторые правила подкрепления базируются на неявных учителях.

Учитывая его состояние и состояние окружающей среды, агент выберет действие, которое максимизирует его вознаграждение или будет исследовать новую возможность. Эти действия изменяют состояние среды и агента. Они также будут интерпретироваться, чтобы дать вознаграждение агенту. Выполняя этот цикл много раз, агенты улучшат свое поведение.

Агент воздействует на среду, а среда воздействует на агента. О такой системе говорят, что она имеет обратную связь. Такую систему нужно рассматривать как единое целое, поэтому линия раздела между средой и агентом достаточно условна.

Основные алгоритмы:

- Q-Learning.

Заключение

Выбор алгоритма зависит от типа задач, которые предстоит решить, а также от доступных данных и алгоритма, по которому будет построена модель. Некоторые виды задач могут поддерживать различные способы машинного обучения, а некоторые только один.

ЛИТЕРАТУРА

1. Айвазян, С.А. Прикладная статистика: основы моделирования и первичная обработка данных / С.А. Айвазян, И.С. Енюков, Л.Д. Мешалкин. – М. : Финансы и статистика, 1983.
2. Айвазян, С.А. Прикладная статистика: исследование зависимостей / С.А. Айвазян, И.С. Енюков, Л.Д. Мешалкин. – М. : Финансы и статистика, 1985.
3. Прикладная статистика: классификация и снижение размерности / С.А. Айвазян [и др.] – М. : Финансы и статистика, 1989.
4. Вапник, В.Н. Восстановление зависимостей по эмпирическим данным / В.Н. Вапник. – М. : Наука, 1979.

УДК 004.8

ЗАДАЧИ, РЕШАЕМЫЕ С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

М.А. ШЕВЦОВ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматриваются задачи, связанные с машинным обучением, которые можно разделить на то, какой вид прогнозирования или предсказания они будут совершать. В этой статье мы рассмотрим основные виды задач, решаемые с помощью машинного обучения.

1. Задача классификации (англ. **classificationproblem**). Имеется множество объектов (ситуаций), разделенных некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

В математической статистике задачи классификации называются также задачами дискриминантного анализа. В машинном обучении задача классификации решается, в частности, с помощью методов искусственных нейронных сетей при постановке эксперимента в виде обучения с учителем.

Типы классов:

- *двухклассовая классификация.* Наиболее простой в техническом отношении случай, который служит основой для решения более сложных задач;
- *многоклассовая классификация.* Когда число классов достигает многих тысяч, задача классификации становится существенно более трудной;
- *непересекающиеся классы;*
- *пересекающиеся классы.* Объект может относиться одновременно к нескольким классам;
- *нечеткие классы.* Требуется определять степень принадлежности объекта каждому из классов, обычно это действительное число от 0 до 1.

Основные задачи:

- определение принадлежности объекта к определенному классу.

2. Кластерный анализ (англ. **clusteranalysis**) – процедура, выполняющая сбор данных, содержащих информацию о выборке объектов, и затем упорядочивающая объекты в однородные группы. Задача кластеризации относится к статистической обработке, а также к широкому классу задач обучения без учителя.

Независимо от предмета изучения применение кластерного анализа предполагает следующие этапы:

- отбор выборки для кластеризации. Подразумевается, что имеет смысл кластеризовать только количественные данные;
- определение множества переменных, по которым будут оцениваться объекты в выборке, то есть признакового пространства;
- вычисление значений той или иной меры сходства (или различия) между объектами;
- применение метода кластерного анализа для создания групп сходных объектов;
- проверка достоверности результатов кластерного решения.

Основные задачи:

- разработка типологии или классификации;
- исследование полезных концептуальных схем группирования объектов;
- порождение гипотез на основе исследования данных;
- проверка гипотез или исследования для определения, действительно ли типы (группы), выделенные тем или иным способом, присутствуют в имеющихся данных.

3. Регрессионный анализ (англ. **regressionanalysis**) – метод моделирования измеряемых данных и исследования их свойств. Данные состоят из пар значений зависимой переменной (переменной отклика) и независимой переменной (объясняющей переменной).

Основные задачи:

- определение степени детерминированности вариации критериальной (зависимой) переменной предикторами (независимыми переменными);
- предсказание значения зависимой переменной с помощью независимой (независимых);

- определение вклада отдельных независимых переменных в вариацию зависимой.

Регрессионный анализ нельзя использовать для определения наличия связи между переменными, поскольку наличие такой связи и есть предпосылка для применения анализа.

Заключение

Каждый тип задач классифицируется по типу решаемых задач, сферы применения и алгоритмам которые поддерживают решение данной проблемы, но важно помнить, что многие задачи разделены весьма условной линией и могут быть взаимозаменяемы в определенных условиях.

ЛИТЕРАТУРА

1. Дрейпер, Н. Прикладной регрессионный анализ. Множественная регрессия = AppliedRegressionAnalysis / Н. Дрейпер, Г. Смит . – 3-е изд. – М. : Диалектика, 2007. – 912 с.
2. Шлезингер, М. Десять лекций по статистическому и структурному распознаванию / М. Шлезингер, В. Главач. – Киев : Наукова думка, 2004.
3. Mitchell, T. Machine Learning / T. Mitchell. – McGraw-Hill Science/Engineering/Math, 1997.
4. Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell (1983), Machine Learning: An Artificial Intelligence Approach, Tioga Publishing Company.

УДК 004.588

**АНАЛИЗ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ,
ПРЕДНАЗНАЧЕННЫХ ДЛЯ ИЗУЧЕНИЯ АНГЛИЙСКОГО ЯЗЫКА****Н.В. ЮРЦЕВИЧ****(Представлено: Т.С. РУДЬКОВА)**

Рассматриваются функциональные возможности наиболее популярных мобильных приложений, предназначенных для изучения английского языка. Приводится обоснование необходимости создания нового мобильного приложения, которое поможет решить актуальную проблему изучения технического английского языка людям, интересующимся сферой информационных технологий.

На сегодняшний день в мире насчитывается более двухсот государств, а языков, на которых говорят их жители, – несколько тысяч. Международные языки позволяют людям общаться друг с другом, независимо от национальности и места проживания. Английский язык является одним из международных языков. Значение английского языка в современном мире настолько велико, что его знание не является привилегией и роскошью. Его учат все и повсеместно: в школах, университетах, на курсах. В век цифровых технологий любой желающий может изучать английский язык с помощью специализированных мобильных приложений. Подразумевается, что любой образованный человек просто обязан владеть английским языком, так как именно он является его ключом к дальнейшему самообразованию и самосовершенствованию. Именно поэтому сейчас существует так много мобильных приложений, предлагающих научить пользователей английскому.

В сфере информационных технологий английский язык является основным языком отрасли. Большинство технической документации и специализированной литературы написано на английском языке. Кроме того, 90% заказчиков программного обеспечения – это компании из Западной Европы или США, и языком коммуникации с ними является английский. Исходя из этих данных можно сделать вывод, что знание английского языка в сфере информационных технологий важно для успешной работы специалиста.

В последние годы сфера мобильных приложений стремительно развивается. Это обусловлено тем, что у подавляющего большинства людей есть современный мобильный телефон, и люди активно устанавливают мобильные приложения. Несомненным плюсом мобильных приложений является то, что мобильный телефон компактный, благодаря этому пользователь может использовать приложение практически везде, где он находится. Исходя из всего этого, объектом исследования в данной статье стали приложения для изучения английского языка пользователями, интересующимися сферой информационных технологий.

На сегодняшний день довольно много приложений, которые бы позволили изучать английский язык, но приложения для специалистов в сфере информационных технологий отсутствуют. Ниже будут рассмотрены популярные бесплатные аналоги с сервиса Google Play [1]. Первым примером рассмотренного приложения является «Английский разговорник», пользовательский интерфейс которого представлен на рисунке 1.

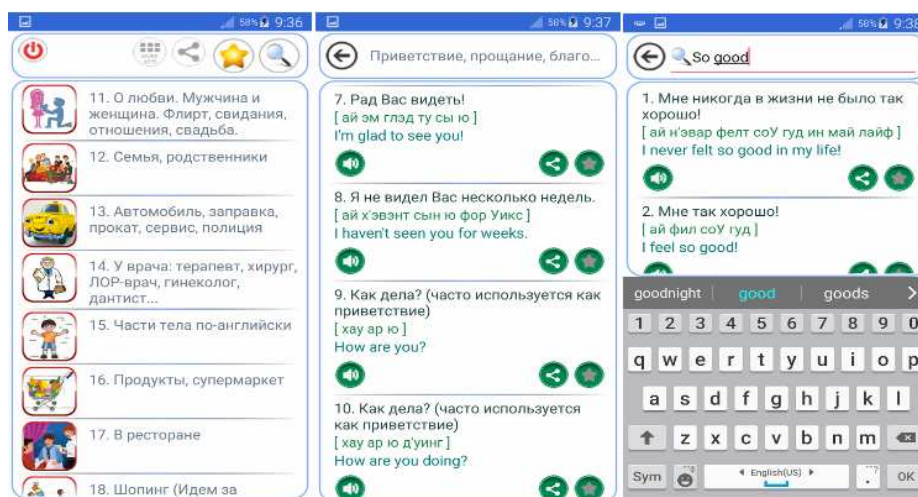


Рисунок 1. – Пользовательский интерфейс мобильного приложения «Английский разговорник»

В приложении представлено 34 категории слов, каждая категория содержит в себя фразы и слова с транскрипцией и произношением – пользователь имеет возможность сохранить слова в категорию избранное, а также реализован поиск по всем категориям. Основное преимущество приложения – не требует подключения к интернету. Также приложение имеет удобный интерфейс и хорошую базу слов. Благодаря этому приложение пользуется популярностью у пользователей (более 100 тыс. установок) и имеет высокий рейтинг (4,7 звезды). Основной недостаток приложения – русскоязычная транскрипция [2].

Вторым примером является приложение «Выучи 90% слов за неделю!», представляющее набор тестов для изучения 300 самых употребляемых английских слов. Также пользователь имеет возможность просмотра статистики. Пример пользовательского интерфейса представлен на рисунке 2.



Рисунок 2. – Пользовательский интерфейс приложения «Выучи 90% слов за неделю!»

Основное преимущество приложения – оно не требует подключения к интернету. Также приложение имеет удобный интерфейс и содержит действительно необходимые слова. Благодаря этому приложение пользуется популярностью у пользователей (более 500 тысяч установок) и имеет высокий рейтинг (4,2 звезды). Основной недостаток приложения – отсутствие категорий [3].

Рассмотренные аналоги не в полной мере отвечают требованиям пользователей, однако при проектировании мобильного приложения для изучения английского языка IT-специалистами можно использовать лучшие практики этих приложений и улучшить их уникальными функциями, которые будут интересны и полезны потенциальным пользователям.

Исходя из рассмотренных выше примеров мобильных приложений, существует необходимость в реализации мобильного приложения со следующим функционалом:

- изучение часто встречающихся слов в технической документации на английском языке;
- выполнение тестовых заданий;
- просмотр статистики (количество изученных слов и категорий, результаты тестов);
- создание, редактирование и удаление пользовательских категорий и прохождение тестов по данным категориям;
- выбор оформления приложения пользователем.

ЛИТЕРАТУРА

1. Google Play // Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Google_Play. – Дата доступа: 11.09.2017.
2. Английский разговорник – Google Play [Электронный ресурс]. – Режим доступа: <https://play.google.com/store/apps/details?id=com.saidjon.ssmphrasebookruen>. – Дата доступа: 12.09.2017.
3. Выучи 90% слов за неделю! – Google Play [Электронный ресурс]. – Режим доступа: <https://play.google.com/store/apps/details?id=com.alexuvarov.android.learntop300englishwords>. – Дата доступа: 12.09.2017.

УДК 004.588

**РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ
ДЛЯ ИЗУЧЕНИЯ АНГЛИЙСКОГО ЯЗЫКА «HELLO-IT-WORLD»****Н.В. ЮРЦЕВИЧ****(Представлено: Т.С. РУДЬКОВА)**

Рассматриваются вопросы разработки мобильного приложения, предназначенного для изучения технического английского языка в сфере информационных технологий.

В мире существует множество стран, а языков, на которых говорят их жители, еще больше. И в данном случае речь идет не о сотне-другой, а о нескольких тысячах. Возникает вопрос, как населению Земли понять друг друга? Этой цели и служат международные языки, которые позволяют всем нам общаться друг с другом, независимо от национальности и места проживания. Один из них – английский язык. Более того, английский язык – язык межнационального общения для всего человечества. Поэтому в настоящее время актуальной является разработка мобильного приложения «Hello-IT-World», которое должно удовлетворять следующим требованиям:

- наличие простого и удобного пользовательского интерфейса;
- наличие обширной базы слов и категорий по IT-тематике;
- наличие тестовых заданий и статистики для каждой категории слов;
- наличие широких функциональных возможностей, а именно: создания и редактирование новых категорий, добавление и редактирование слов в созданные категории, добавление тестовых заданий и статистики по созданным категориям.

При разработке гибридного мобильного приложения использовалась платформа Apache Cordova. Она позволила использовать стандартные веб-технологии, такие как HTML, CSS и JavaScript для кросс-платформенной разработки, избегая родного языка разработки для каждой из мобильных платформ.

HTML был использован для создания каркаса приложения. HTML (от англ. HyperText Markup Language – «язык гипертекстовой разметки») – стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства [1].

CSS был использован для оформления внешнего вида приложения. CSS (англ. Cascading Style Sheets – каскадные таблицы стилей) – формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL [2].

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный и функциональный стили. Он был использован для написания функциональной части приложения [3].

Web SQL Database использовался для доступа к индексируемым базам данных.

Web SQL Database API не является частью спецификации HTML5, это отдельная спецификация, которая вводит набор программных интерфейсов для управления клиентской базой данных с помощью языка SQL. Существуют три основных метода, определенных в данной спецификации:

- 1) OpenDatabase – метод создает объект базы данных, либо использует существующую базу данных;
- 2) Transaction – этот метод дает нам возможность контролировать действия выполнения или отката запроса в зависимости от ситуации;
- 3) ExecuteSQL – метод используется для выполнения фактического SQL запроса [4].

При разработке программного обеспечения разработана функциональная структура, которая состоит из следующих подсистем:

- *подсистема изучения слов* – отвечает за корректное отображение категорий и слов, извлекаемых из базы данных приложения, предназначена для изучения пользователями написания и произношения английских слов;
- *подсистема прохождения тестов* – обеспечивает генерацию и корректное отображение тестовых заданий для каждой категории слов с возможностью выбора пользователем направления перевода;
- *подсистема сбора статистики* – предоставляет возможность сбора и отображения статистической информации;
- *подсистема поиска слов* – предоставляет пользователю возможность поиска слов по базе данных приложения;

- подсистема добавления категорий и слов – позволяет пользователю приложения создавать новые категории и добавлять в них английские слова с переводом, а также редактировать и удалять их;
- подсистема настройки приложения – позволяет пользователю настраивать внешний вид мобильного приложения.

Функциональная структура системы представлена на рисунке 1.

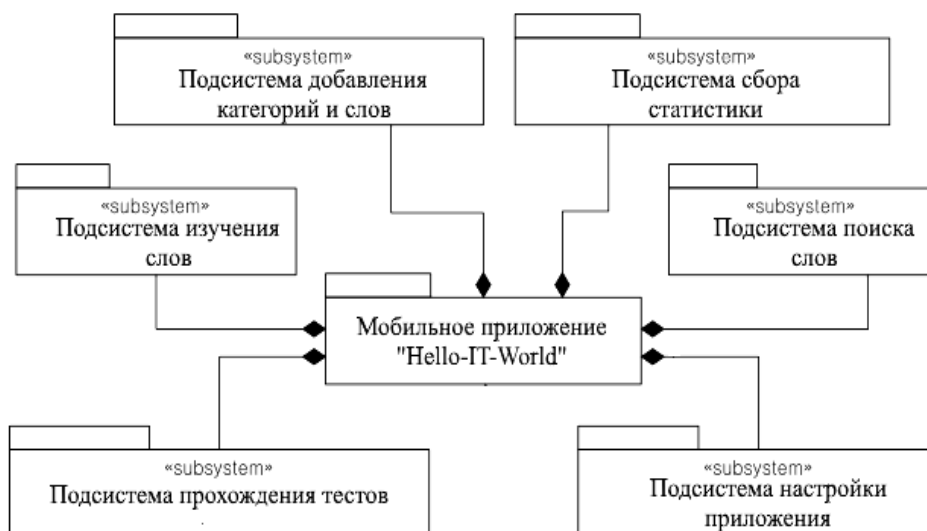


Рисунок 1. – Функциональная структура приложения

Подсистема прохождения тестов состоит трех частей, таких как:

- выбор категории и направления перевода, за что отвечает функция `initCategoryList`;
- автоматически генерируемый тест, за генерацию которого отвечает функция `startTest`;
- окна с результатами прохождения теста, за отображение которого отвечает функция `rez`.

При нажатии на выпадающий список пользователь может выбрать желаемую категорию для прохождения теста. Также с помощью флажков, пользователь может выбрать направления перевода слов. После нажатия кнопки «Начать тест», тест будет автоматически сгенерирован.

В автоматически сгенерированном тесте учитывается выбор категории и направление перевода, указанные пользователем ранее. Количество вопросов в тесте равно количеству слов в категории. В каждом вопросе есть четыре варианта ответа, и только один правильный. Основное преимущество генерируемого теста в том, что при каждом новом запуске теста варианты ответа расположены в абсолютно случайном порядке, и при этом в одном вопросе не может быть двух одинаковых вариантов ответа.

При прохождении теста пользователь в любой момент может вернуться к настройкам теста, благодаря кнопке «Назад» (функция `onStart`). Для пролистывания теста в нижней части экрана появляется кнопка «Вверх» (функция `scroll`), при нажатии на которую пользователь в любой момент может вернуться к началу теста. Выбор ответа осуществляется нажатием на флажок напротив вариантов ответа. Для удобства пользователя, на тот случай, если он пропустит несколько вопросов при прохождении теста, по умолчанию флажок установлен на нулевой (скрытый) вариант ответа.

После ответа на все вопросы пользователь должен нажать кнопку «Результаты теста» тест закроется, и пользователь увидит статистику своих ответов (рис. 2).

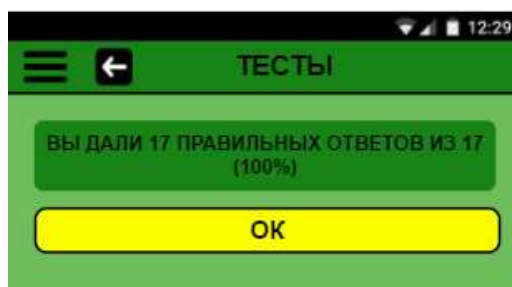


Рисунок 2. – Результаты прохождения теста

Результаты прохождения теста влияют на рейтинг слов. Изначально слово имеет рейтинг 0, при правильном ответе на вопрос теста рейтинг увеличивается на 1, при неправильном ответе – уменьшается на 1. Также процент правильных ответов влияет на рейтинг категории. Изначально категория имеет рейтинг 0. Если пользователь дает от 68 до 100% правильных ответов, рейтинг увеличивается на 1, если правильных ответов от 34 до 67% – рейтинг категории не меняется, если же пользователь дает менее 34% правильных ответов, рейтинг категории уменьшается на 1. Границы рейтинга слов и категорий от 0 до 5. Графически рейтинги отображаются около слов и категорий в словаре и статистике с помощью звезд, число желтых звезд численно равно рейтингу слова или категории.

После того как был написан функционал мобильного приложения, оно было протестировано, чтобы удостовериться в том, что программный продукт соответствует указанным ранее требованиям. Тесты выявили несколько дефектов приложения, которые были незамедлительно устранены. Повторное тестирование дефектов не выявило.

Результатом проделанной работы стало мобильное приложение для изучения английского языка. Разработанный программный продукт может использоваться для изучения английского языка пользователями, которые планируют приобретать и совершенствовать навыки английского языка для последующего применения полученных знаний в сфере информационных технологий.

ЛИТЕРАТУРА

1. HTML [Электронный ресурс] // Википедия, свободная энциклопедия. – Электронный ресурс. – Режим доступа: <https://ru.wikipedia.org/wiki/HTML>. – Дата доступа: 23.09.2017.
2. CSS [Электронный ресурс] // Википедия : свободная энцикл. – Электронный ресурс. – Режим доступа: <https://ru.wikipedia.org/wiki/CSS>. – Дата доступа: 23.09.2017.
3. JavaScript [Электронный ресурс] // Википедия, свободная энцикл. – Электронный ресурс. – Режим доступа: <https://ru.wikipedia.org/wiki/JavaScript>. – Дата доступа: 23.09.2017.
4. HTML5 Web SQL Database [Электронный ресурс] // Web онлайн учебники. – Электронный ресурс. – Режим доступа: <http://www.w3big.com/ru/html/html5-web-sql.html>. – Дата доступа: 24.09.2017.

УДК 004.588

ИСПОЛЬЗОВАНИЕ APACHE CORDOVA КАК ПЛАТФОРМЫ ДЛЯ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ В СВЯЗКЕ С ФРЕЙМВОРКОМ IONIC

Д.В. ДРОЗД**(Представлено: Д.В. ПЯТКИН)**

Рассматривается использование Apache Cordova и Ionic веб-разработчиками для нескольких платформ с единой реализацией, позволяющей использовать стандартные веб-технологии, такие как HTML5, CSS3 и JavaScript, для кроссплатформенной разработки, избегая родного языка разработки для каждой из мобильных платформ.

При разработке мобильного приложения требуется отдельная реализация для каждой мобильной платформы. В связи с этим усложняется разработка мобильного приложения, привлекая для проекта большое количество разработчиков. Использование *Apache Cordova* позволяет свести штат разработчиков к минимуму, минимизировать финансовые и временные затраты на разработку и позволяет включить в разработку мобильного приложения веб-разработчиков. В Фреймворк *Ionic* входит набор инструментов и виджетов для создания приложений. Это анимированные всплывающие окна на странице, кнопки, слайдеры и многое другое.

Данная работа поможет получить представление о платформе *Apache Cordova* и Фреймворке *Ionic*, что хорошо подходит для кроссплатформенной разработки мобильных приложений.

Apache Cordova – это платформа разработки мобильных приложений с открытым исходным кодом, которая позволяет использовать стандартные веб-технологии, такие как HTML5, CSS3 и JavaScript для кроссплатформенной разработки, избегая родного языка разработки для каждой из мобильных платформ. Приложения выполняются внутри обертки нацеленной на каждую платформу и полагаются на стандартные API для доступа к датчикам устройства, данным и состоянию сети.

На сегодняшний день самой популярной оболочкой для гибридных приложений является *Apache Cordova* (ранее – *PhoneGap*). Это платформа, которая помимо оборачивания кода в *WebView* предоставляет огромный набор плагинов js API, с помощью которых можно полноценно взаимодействовать с операционной системой и устройствами смартфона. У *Apache Cordova* большое сообщество, множество готовых решений и плагинов, но для полноценного приложения не хватает красивого, современного UI, который бы придал приложению удобство управления и ощущение, что это приложение, а не адаптированный сайт.

Ionic – это Фреймворк, с хорошо проработанным графическим интерфейсом, стилизованный под различные платформы для максимальной схожести с нативными приложениями. За основу у Фреймворка взяты *AngularJS* и *SASS*, что принесло ему большую популярность у разработчиков, так как эти технологии сводят порог вхождения к минимуму. Ядром фреймворка является *Apache Cordova*, поэтому все плагины *Apache Cordova* будут отлично работать на *Ionic*.

Чтобы начать работать с *Apache Cordova* и *Ionic* требуется установить несколько настроек для библиотеки и Фреймворка:

- 1) установка пакетного менеджера *node.js* – *npm*. С его помощью можно управлять модулями и зависимостями;
- 2) установка *Node.js* – программная платформа, превращающая JavaScript из узко специализированного языка в язык общего назначения;
- 3) установка *git* клиента;
- 4) установка модуля *cordova* с помощью инструмента *npm* (листинг 1);
- 5) установка модуля *ionic* с помощью инструмента *npm* (листинг 2);
- 6) создание приложения через *ionic* фреймворк (листинг 3);
- 7) добавить платформы в *Apache Cordova*, для которых требуется разработка мобильного приложения (листинг 4);
- 8) борка исходного файла для мобильного приложения (листинг 5).

Листинг 1 – Установка *Apache Cordova*

```
npm install -g cordova
```

Листинг 2 – Установка Фреймворка ionic

```
npm install -g ionic
```

Листинг 3 – Создание нового мобильного приложения

```
ionic start JoinUs
```

Листинг 4 – Добавление платформы android в Apache Cordova

```
cordova platform add android
```

Листинг 5 – Сборка apk мобильного приложения для android устройств

```
ionic cordova build – release android
```

Заключение

В данной работе рассмотрены возможности создания и экспорта веб-приложения в мобильные приложения с использованием платформы *Apache Cordova* и *Ionic* Фреймворк.

Показан базовый пример по экспорту веб-приложения в мобильные устройства на операционной системе *Android*, демонстрирующий простоту применения данных технологий в рамках кроссплатформенной разработки.

ЛИТЕРАТУРА

1. Apache Cordova [Электронный ресурс] Интерфейс командной строки. – Режим доступа: <https://cordova.apache.org/docs/ru/latest/guide/cli/>. – Дата доступа: 25.09.2017.
2. Ionic Framework [Электронный ресурс] Publishing your app. – Режим доступа: <http://ionicframework.com/docs/v1/guide/publishing.html>. – Дата доступа: 25.09.2017.
3. Блог Front-End Developer'a [Электронный ресурс] Гибридные приложения, разработка с Ionic. – Режим доступа: <https://golosay.net/hybrid-apps-developing/>. – Дата доступа: 25.09.2017.

УДК 004.588

**РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ «JOINUS»,
ПРЕДНАЗНАЧЕННОГО ДЛЯ ОРГАНИЗАЦИИ РАЗВЛЕКАТЕЛЬНЫХ
И КУЛЬТУРНО-ОБЩЕСТВЕННЫХ МЕРОПРИЯТИЙ**

Д.В. ДРОЗД

(Представлено: Д.В. ПЯТКИН)

Рассматривается мобильное приложение, предоставляющее возможность пользователям создавать свои объявления о мероприятиях, а также найти пользователю мероприятие по интересу. Представлена разработка приложения «JoinUs».

Современный смартфон или планшет способен полностью заменить весь функционал домашних компьютеров, в связи с чем актуальность приобретает разработка приложений не столько для персональных компьютеров, как для планшетов и смартфонов. С популяризацией мобильных устройств пользователи чаще ищут необходимую информацию на своих мобильных устройствах. Вследствие этого актуальность разработки мобильного приложения возрастает, а вместе повышается актуальность в поисках или организации мероприятий прямо с мобильного устройства.

Разработка данного мобильного приложения «JoinUs», предназначенного для организации развлекательных и культурно-общественных мероприятий, является реализацией мобильного приложения, предоставляющего возможность пользователям создавать свои объявления о мероприятиях, а также найти пользователю мероприятие по его интересу.

Приложение не только должно найти заинтересовавшее пользователя мероприятие, но и предоставить ему всю необходимую информацию об этом событии: дату, место, описание и многое другое. Ознакомительная информация о мероприятии должна быть представлена в простой и понятной форме в виде текста и местоположения на карте. Процесс загрузки, установки, использование должен быть максимально удобен.

Целевой аудиторией мобильного приложения являются любые пользователи, проживающие на территории Республики Беларусь.

Для разработки данного мобильного приложения используется библиотека *Apache Cordova* с *Ionic* фреймворком, данная связка позволяет разработать кроссплатформенное мобильное приложение.

Для дизайна мобильного приложения используется *Material Design*. *Material Design* – дизайн программного обеспечения и приложений от компании Google. Идея дизайна заключается в приложениях, которые открываются и сворачиваются как карточки, используя эффекты теней.

Разработка данного мобильного приложения написано на языке программирования JavaScript с использованием AngularJS Framework, а также современных средств для работы с веб, такими как HTML5, CSS3, JavaScript, Google Maps API.

В качестве сервера выступает *Cloud9*. *Cloud9* – интегрированная среда разработки, предоставляемая по модели облачных вычислений. Среда поддерживает несколько языков программирования, при этом, основной фокус направлен на JavaScript, HTML и CSS. Среда разработки полностью написана на JavaScript и использует Node.js на серверной стороне. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API.

Вся информация на сервере хранится в *MongoDB*. *MongoDB* – документоориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы. Пример данных в MongoDB в виде JSON-подобных документов представлен в листинге 1.

Листинг 1 – Описание данных в MongoDB о пользователе и мероприятии

```
_id: "58b97f69a20c1217adfd9a4f",  
username: "deniskogo",  
firstname: "Denis",  
lastname: "Drozd",  
avauser: "data:image/png;base64..."
```

```
_id: "595485da9becc1068b93b546",  
author: "58b97f69a20c1217adfd9a4f",  
header: "Пробное",  
text: "Немного текста",  
date: "2017-06-29T21:00:00.000Z",  
dateCreated: "2017-08-29T09:45:11.188Z",  
x: 55.49285609375866,  
y: 28.77653863769524,  
likeUsers:  
[  
    "58f8764b8112a10877a010ef",  
    "58c0ffda03e8990c585d0127"  
]
```

Мобильное приложение «JoinUs» состоит из множества страниц, каждая из них отвечает за свою задачу:

1. Страница аутентификации пользователя.
2. Страница регистрации пользователя.
3. Персональная страница пользователя.
4. Боковое меню.
5. Список мероприятий.
6. Страница сообщений пользователя.

Заключение

В результате выполненной работы реализовано мобильное приложение «JoinUs» предназначенное для организации развлекательных и культурно-общественных мероприятий. Для мобильного приложения разработан удобный для работы интерфейс, который не требует дополнительного обучения и интуитивно понятен, а также был выделен сервер для хранения необходимой информации для мобильного приложения.

ЛИТЕРАТУРА

1. Магазин мобильной электроники НОУ-ХАУ [Электронный ресурс] Нужен ли планшет в «мире смартфонов». Актуальность планшетов в 2017. 19 июня 2017. Новости. Обзоры и статьи о технике. – Режим доступа: <http://nou-hau.pf/article/aktualnost-planshetov-v-2017>. – Дата доступа: 25.09.17.
2. Википедия [Электронный ресурс] AngularJS. – Режим доступа: <https://ru.wikipedia.org/wiki/AngularJS>. – Дата доступа: 25.09.17.
3. Википедия [Электронный ресурс] Material Design. – Режим доступа: https://ru.wikipedia.org/wiki/Material_Design. – Дата доступа: 25.09.2017.
4. Apache Cordova [Электронный ресурс] Интерфейс командной строки. – Режим доступа: <https://cordova.apache.org/docs/ru/latest/guide/cli/>. – Дата доступа: 25.09.2017.
5. Ionic Framework [Электронный ресурс] Publishing your app. – Режим доступа: <http://ionicframework.com/docs/v1/guide/publishing.html>. – Дата доступа: 25.09.2017.
6. Learn Javascript [Электронный ресурс] Скринкаст по Node.JS. – Режим доступа: <https://learn.javascript.ru/screencast/nodejs>. – Дата доступа: 25.09.2017.
7. Википедия [Электронный ресурс] Cloud9. – Режим доступа: <https://ru.wikipedia.org/wiki/Cloud9>. – Дата доступа: 25.09.2017.

УДК 004.45

**АНАЛИЗ ФУНКЦИОНАЛЬНОСТИ ЭЛЕКТРОННЫХ ПУТЕВОДИТЕЛЕЙ
НА ПРИМЕРЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ И WEB-САЙТА****А.В. ДЫСИН****(Представлено: Д.В. ПОПКОВА)**

Рассмотрено понятие «путеводитель», виды путеводителей. Изучены функции путеводителей. Проведено исследование характеристик web-приложения и web-сайта с точки зрения функциональности. Выделены плюсы и минусы функциональных возможностей программных средств.

Путеводитель – краткое справочное издание. Путеводители по стране, городу и т.п., предназначенные главным образом для туристов, содержат сведения о местных достопримечательностях, учреждениях культуры, путях сообщения, отелях и другую информацию. Широко известны «путеводители по странам – Бедекеры». Путеводители по отдельным учреждениям культуры (музеям, крупным библиотекам) или исторически-художественным комплексам сообщают об их истории, структуре, фондах.

Существует несколько функций путеводителей, которые позволяют раскрыть их сущность в современном мире.

Путеводитель как вид справочного издания

Путеводитель как справочник представляет собой сложно организованную, эклектичную структуру, в которой соединена научно-популярная информация комплексного характера и информация бытового плана, не претендующая на научность и строгость. В зависимости от структуры и содержания текста, формируются различные виды «взаимодействия» читателя с текстом: предварительное и выборочное ознакомление; последовательное, «академическое» изучение; прерывистое чтение, совмещенное с перемещением в пространстве и осмотром достопримечательностей [1].

Трансляция научного знания на уровень обыденного сознания. Путеводитель можно рассматривать как текст, соединяющий научно-справочный и научно-популярный подстили. Будучи текстом, созданным для «культурной рекреации», Бедекер, по сравнению с другими разновидностями научно-популярного жанра, значительно упрощает, редуцирует, унифицирует и эмоционально окрашивает сообщаемую читателю информацию. В наибольшей степени это проявляется в характере введения в текст специальных терминов, в особенностях цитирования, в расширенном использовании эпитетов, гипербола и метафор.

Путеводитель как руководство по формированию новых поведенческих стереотипов. В самом жанре путеводителя, рожденного культурой, как ответ на массовый запрос в руководстве, априорно заложена функция воздействия на читателя. Используемые приемы руководства достаточно разнообразны: от демонстрации престижности тех или иных действий до советов, подбадриваний, понуждений и даже императивов. Путеводитель контролирует также визуальное восприятие: не только отбирая объекты или степень детализации при осмотре, но и предписывая «соответствующие» реакции и впечатления. Текст диктует распределение времени, характер, скорость, ритм передвижения человека в туристском пространстве [1].

Рекламно-имиджевые функции путеводителя. В туристском пространстве регион выступает своеобразным «товаром» и задача путеводителя – привлечь к нему внимание. Основные приемы, используемые для реализации данной функции – позиционирование, персонификация имиджа, визуализация, преувеличение, сопоставление с всемирно известными достопримечательностями и т.д.

Путеводитель как инструмент конструирования достопримечательностей

Конструирование достопримечательностей включает ряд операций: переводение памятника из поля функционирования искусства в поле функционирования массовой культуры; упрощение, а порой и искажение смыслов, ценностей, значений; тиражирование и организация массового потребления; переводение из сферы незаинтересованного эстетического восприятия в сферу престижного потребления [1].

Электронный путеводитель – широкое понятие.

Он может иметь разные формы:

1. *Аудиогид* – воспроизводится на любом устройстве, поддерживающем mp3. *Аудиоэкскурсия* – это маршрут, состоящий из остановок, каждой из которых соответствует один трек.
2. *Электронный буклет* – на экране воспроизводится файл буклета, с картой и текстом (обычно в pdf-формате).
3. *Интерактивный справочник* – на экране воспроизводится интерфейс программы с доступом к карте и справочной информации.
4. *Мультимедийный путеводитель* – на карте размещены гипертекстовые ссылки, открывающие фотографии, описание, звуковой комментарий, видеоролик.

5. *Мультимедийный навигатор* (GPS-путеводитель) – определяет местонахождение экскурсанта, показывает дорогу и автоматически включает звуковой комментарий [3].

Следует различать так называемые «умные путеводители», которые сами могут определить, где находится человек, и путеводители с ручным управлением.

Под открытым небом используют путеводители, имеющие спутниковую навигацию. Сейчас действует американская система GPS и российская ГЛОНАСС. Однако сила сигнала спутниковой навигации очень мала – в помещениях GPS не действует.

Задача автоматического ориентирования внутри помещений решается одним из *методов альтернативной навигации*:

1. *Беспроводные сети (WiFi)*. Принцип работы: в здании устанавливается сервер и беспроводные точки доступа (WirelessAccessPoint). Программа считывает через точки доступа информацию, хранящуюся в базе данных сервера. Плюсы – не надо хранить много информации, работает быстрее; базу данных можно обновлять централизованно. Минусы – дорого, сложная технология.

2. *Навигация по мобильной связи (GSM)*. Местоположение определяется по близости к базовой станции сотовой связи.

3. *Радиочастотная идентификация (RFID)*. Навигация ведется по специальным RFID-меткам. Это чипы, которые прикрепляются к объектам. Приемник считывает информацию с этих чипов. Такая система активно используется в логистике, на больших складах, в транспорте.

Очевидно, такие проекты требуют немалых затрат, поэтому создаются путеводители с ручным управлением.

Электронный путеводитель может быть в виде Web-сайта или программы.

Web-сайт – это объединенная под одним адресом (доменным именем) совокупность документов юридического или физического лица. По умолчанию подразумевается, что сайт располагается в сети Интернет, но он также может публиковаться в локальной сети компании – в сети Интранет [1].

Интернет-сайты или онлайн-путеводители обновляются быстрее и чаще, чем их аналоги, поэтому в них можно найти больше заслуживающей доверия информации, чем в книгах самого авторитетного издательства. На таких сайтах размещаются GPS-маршруты экскурсии, информация о достопримечательностях и их фотографии. Достоинством таких путеводителей является бесплатная информация, постоянные обновления, нет необходимости хранить много информации; недостатком – наличие доступа в Интернет, большие объемы загружаемой информации [1].

Программа устанавливается на мобильное устройство (КПК, смартфон, iPod, автонавигатор), персональный компьютер (ноутбук) или запускается с диска. Достоинством является автономность, приспособленность для конкретной цели и устройства. Недостатком – программа обычно платная, данные могут устаревать, не всегда хорошо устанавливается, бывают сбои.

В последнее время популярными становятся информационные сенсорные киоски с электронными системами навигации. Такие справочники-путеводители устанавливаются преимущественно в музеях, торговых, бизнес- и публичных центрах.

Пример мобильного приложения «Redigo». Это – компактный гид по странам мира от Австралии до Японии. Он содержит следующую информацию: общая информация о стране, информация о местах и городах, которые рекомендуется посетить, оффлайн карты, где отмечены достопримечательности, информация об актуальных событиях. На рисунке 1 представлен внешний вид приложения «Redigo».

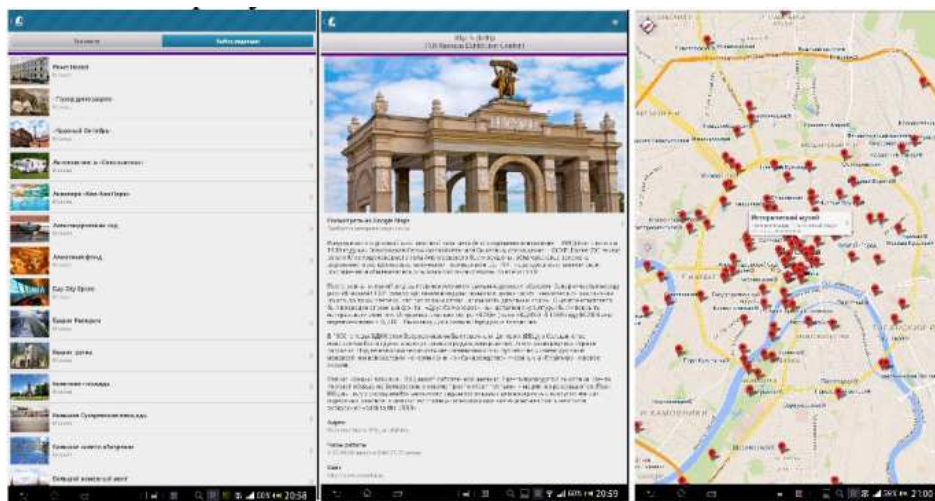


Рисунок 1. – Вид сайта «Redigo»

Приложение Redigo – это гид по большому количеству стран и городов. Информация о каждом городе скачивается на устройство отдельно. В данном примере разберем функциональность приложения на примере города Санкт-Петербурга.

Функциональные возможности приложения Redigo:

- поиск достопримечательностей города;
- поиск достопримечательностей посредством геолокации;
- наличие разговорника на таких языках, как английский, испанский, итальянский, китайский, немецкий и французский;

- наличие разговорника в картинках;

- поиск конкретной достопримечательности по названию;
- возможности скачать оффлайн-карту города либо ее онлайн просмотр.

Однако у данного приложения имеются следующие недоработки:

- несистематизированная информация о достопримечательностях города;
- отсутствие информации об объектах размещения;
- отсутствие информации о предприятиях питания;
- отсутствие возможности проложить маршрут;
- для загрузки оффлайн-карты необходим доступ в интернет.

Примеры Web-сайта – «Путеводитель по Санкт-Петербургу». Это – web-сайт путеводитель по Санкт-Петербургу, который служит помощником не только для туристов, но и для жителей города. На сайте представлена общая информация о городе, о существующих экскурсиях по Санкт-Петербургу и пригородам. В разделе «Карта» отмечены главные достопримечательности города, а также средства размещения, предприятия питания, объекты здравоохранения, объекты досуга и развлечений, транспортного сообщения, последние новости города.

Внешний вид сайта «Путеводитель по Санкт-Петербургу» (<http://cityguidespb.ru>) представлен на рисунке 2.

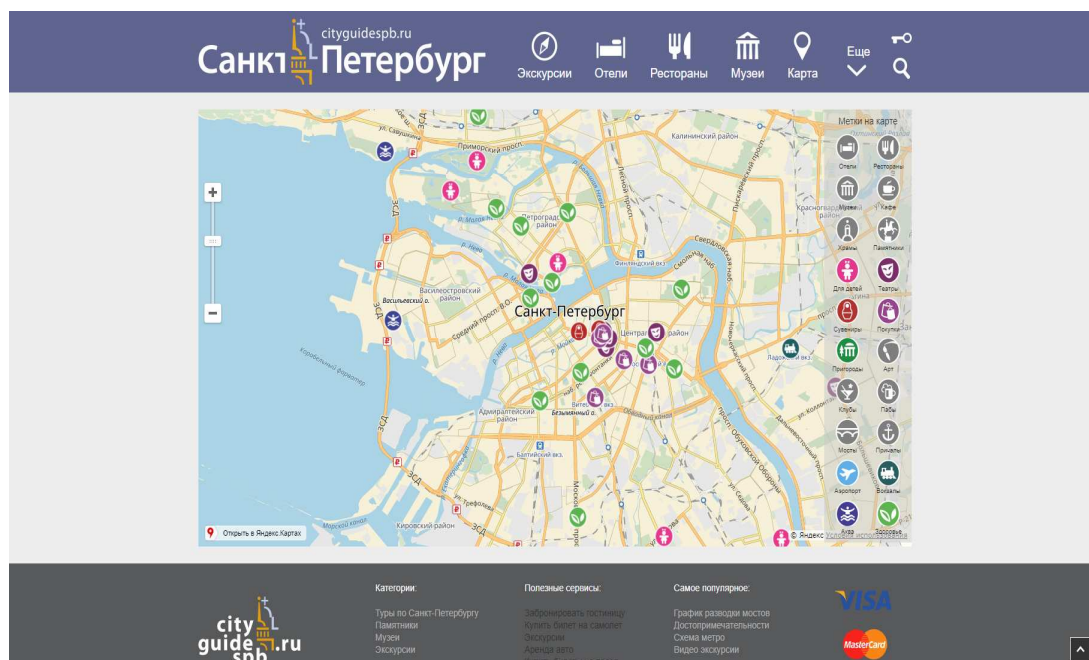


Рисунок 2. – Вид сайта «Путеводитель по Санкт-Петербургу»

Функциональные возможности Web-сайта «Путеводитель по Санкт-Петербургу»:

- предоставление актуальной информации о городе;
- поиск объектов города таких как: объекты размещения, предприятия питания, музеи и т.д.;
- поиск конкретной достопримечательности по названию;
- предоставление информации об экскурсиях и турах Санкт-Петербурга;
- предоставление информации о торговых комплексах города;
- предоставление информации о маршрутах общественного транспорта следующих до необходимого объекта Санкт-Петербурга;

- возможности оставить отзыв;
- реализации личного кабинета пользователя;
- предоставление видео экскурсий.

Недостатки web-сайта:

- для работы с сайтом необходим постоянный доступ к интернету;
- отсутствие возможности проложить маршрут до объекта города на карте;
- отсутствие информации о расстоянии до искомого объекта Санкт-Петербурга;
- большие объемы загружаемой информации.

Таким образом, на основе анализа выбранных источников, можно сделать следующие **выводы:** приложение Redigo как источник информации о странах, за исключением возможности использования-оффлайн, является менее функциональным, так как имеет только общую информацию о достопримечательностях города и страны и не может выступать подробным путеводителем. В то время как Web-сайт «Путеводитель по Санкт-Петербургу» имеет полную информацию обо всех необходимых для путешественника объектах и более полный функционал.

На основе данных выводов выявлена лучшая функциональность у тех видов Web-приложений и Web-сайтов, которые предназначены для узкого направления, так как в них можно наиболее полно отобразить интересующую информацию и сделать расширенные функциональные возможности.

Учитывая все плюсы и минусы рассмотренных в данной работе web-приложения и web-сайта, можно говорить о необходимости разработать улучшенную модель путеводителя, которая будет ориентирована на один город и включать в себя всю необходимую функциональность.

ЛИТЕРАТУРА

1. Сайт – Википедия [Электронный ресурс]. – Режим доступа: <http://ru.wikipedia.org/wiki/Сайт>. – Дата доступа: 10.09.2017.
2. Веб-сайты: определение и применение [Электронный ресурс]. – Режим доступа: <http://www.webtec.com.ua/articles/index/view/web-site/>. – Дата доступа: 13.09.2017.
3. Электронные путеводители: Что это и с чем их едят? [Электронный ресурс]. – Режим доступа: <https://www.skyscanner.ru/news/elektronnye-putevoditeli-chto-eto-i-s-chem-ikh-edyat>. – Дата доступа: 15.09.2017.

УДК004.924

ОСОБЕННОСТИ РЕАЛИЗАЦИИ ИНТЕРАКТИВНОГО 3D-ПУТЕВОДИТЕЛЯ ПО ГОРОДУ ПОЛОЦКУ

А.В. ДЫСИН**(Представлено: Д.В. ПОПКОВА)**

Рассмотрена программная библиотека для языка JavaScriptWebGraphicsLibrary. Выделены ее основные достоинства и недостатки, проанализирована поддержка библиотеки существующими браузерами. Рассмотрены основные библиотеки и фреймворки для разработки на WebGL. Разработан интерактивный 3D-путеводитель по городу Полоцку.

Web-технологии прочно вошли в повседневную жизнь современного человека. Просмотр новостей, совершение покупок, общение и работа в большинстве случаев осуществляются с помощью всемирной сети. Индустрия услуг и развлечений в сети Интернет стремительно развивается, ведущие разработчики программного обеспечения улучшают поддержку трехмерной графики в своих продуктах. Традиционно ее поддержка ограничивалась высокопроизводительными компьютерами или специализированными игровыми консолями, а программирование требовало применения сложных алгоритмов. Однако благодаря росту производительности персональных компьютеров и расширению возможностей браузеров, стало возможным создание и отображение трехмерной графики с применением Web-технологий.

WebGL (WebGraphicsLibrary) – программная библиотека для языка JavaScript, предназначенная для визуализации интерактивной трехмерной графики и двухмерной графики в пределах совместимости Web-браузера без использования плагинов. Одно из преимуществ WebGL – приложения конструируются как Web-страницы, то есть одна и та же программа будет успешно выполняться на самых разных устройствах (к примеру, на смартфонах, планшетных компьютерах и игровых консолях) [1].

С развитием языка гипертекстовой разметки HTML разработчики получили возможность создавать все более сложные Web-приложения. Изначально HTML предлагал только возможность отображения статического контента, но с добавлением поддержки языка JavaScript стало возможным реализовывать более сложные взаимодействия элементов и отображения динамического контента. Внедрение стандарта HTML5 позволило использовать новые возможности, включая поддержку двухмерной графики в виде тега canvas. Создание технологии WebGL позволило отображать и манипулировать трехмерной графикой на Web-страницах с помощью JavaScript. При помощи WebGL разработчики могут создавать совершенно новые пользовательские интерфейсы, трехмерные игры и использовать трехмерную графику для визуализации различной информации. Несмотря на внушительные возможности, WebGL отличается от других технологий доступностью и простотой использования.

В настоящий момент WebGL поддерживается браузерами:

1. *Десктопные браузеры:*
 - MozillaFirefox(с 4-й версии);
 - GoogleChrome (с 9-й версии);
 - Safari (с 6-й версии, по умолчанию поддержка WebGL отключена, чтобы ее включить пользователю необходимо поставить соответствующий флаг в инструментах разработчика);
 - Opera (с 12-й версии, по умолчанию также отключена поддержка WebGL);
 - IE (с 11-й версии, для других версий можно использовать сторонние плагины, например, IEWebGL).

2. *Мобильные браузеры:*

- Android-браузер (поддержка WebGL только на нескольких устройствах);
 - OperaMobile (начиная с 12-й версии, и только для OSAndroid);
 - IOS (полная поддержка с версии 8.1);
 - Firefox for mobile (с 4-й версии);
 - Google Chrome для Android (с 25-й версии) [2].
- Преимущества использования WebGL:
- кроссбраузерность и отсутствие привязки к определенной платформе;
 - использование языка JavaScript, который достаточно распространен;
 - автоматическое управление памятью. В отличие от OpenGL, в WebGL не надо выполнять специальные действия для выделения и очистки памяти;
 - поскольку WebGL для рендеринга графики использует графический процессор на видеокарте (GPU), для этой технологии характерна высокая производительность, которая сравнима с производительностью нативных приложений.

Недостатки использования WebGL:

- отсутствие полной поддержки во всех браузерах;

- медленная работа на мобильных устройствах [3].

Технология WebGL использует низкоуровневое API, что облегчает разработчикам браузеров внедрять ее в свои продукты, но создает достаточно большие трудности при создании интерфейсов. Для упрощения работы разработчиков сайтов были созданы библиотеки, фреймворки и сторонние программные средства.

Библиотеки и фреймворки для разработки на WebGL:

1. *Библиотека WebGLU*. Набор утилит низкого и высокого уровня для разработки приложений на WebGL. WebGLU сконструирован таким образом, что разработчик может сосредоточиться на конечном результате, но библиотека не ограничивает разработчиков, которые хотят больше контроля над системой разработки. Это достигается за счет возможности использования низкоуровневых функций для работы с WebGL API.

2. *Библиотека GLGE*. Библиотека для разработки приложений с использованием WebGL. Библиотека ориентирована больше на динамическое изменение сцены. Однако последние изменения в репозитории осуществлялись в 2014 году, поэтому есть причины усомниться в актуальности библиотеки на сегодняшний день.

3. *Библиотека Three JS*. Наиболее популярная и активно развивающаяся библиотека на сегодняшний день. Подробная и доступная документация и огромное количество рабочих примеров делают эту библиотеку одним из лидеров среди аналогичных систем.

4. *Библиотека Babylon JS*. Библиотека с открытым исходным кодом для создания полноценных 3D-приложений и игр, работающих в веб-браузере без использования сторонних плагинов и расширений. Babylon JS по своим возможностям близок к ThreeJS, однако имеет в своем арсенале некоторые встроенные функции, недоступные в Three JS из коробки. К таким приятным особенностям относятся встроенный физический движок oimo.js – достаточно простой способ создать реалистичный ландшафт, используя карту высот. Разумеется, в threejs также присутствуют такие возможности, но реализованы они при помощи различных дополнительных приложений. Однако за функциональность библиотеки приходится платить нескромным весом в 800 кб.

На основе данного исследования был разработан интерактивный 3D-путеводитель по городу Полоцку с использованием технологии WebGL.

3D-путеводитель состоит из двух функциональных частей: основной клиентской части сайта и его административной панели. Перед полной загрузкой главной страницы отображается анимированный предзагрузчик. Главная страница клиентской части сайта представлена на рисунке 1.



Рисунок 1. – Главная форма интерактивного путеводителя

При выборе необходимого района осуществляется анимированный переход к выбранной локации с отображением всех существующих объектов города. На странице справа находится поле для поиска предприятий размещения и питания, достопримечательностей, а также торговых помещений с сувенира-

ми. Отображение результатов поиска можно выбрать как по алфавитному порядку, так и по классификации, представленной на рисунке выше. Страница отдельного района Полоцка представлена на рисунке 2.

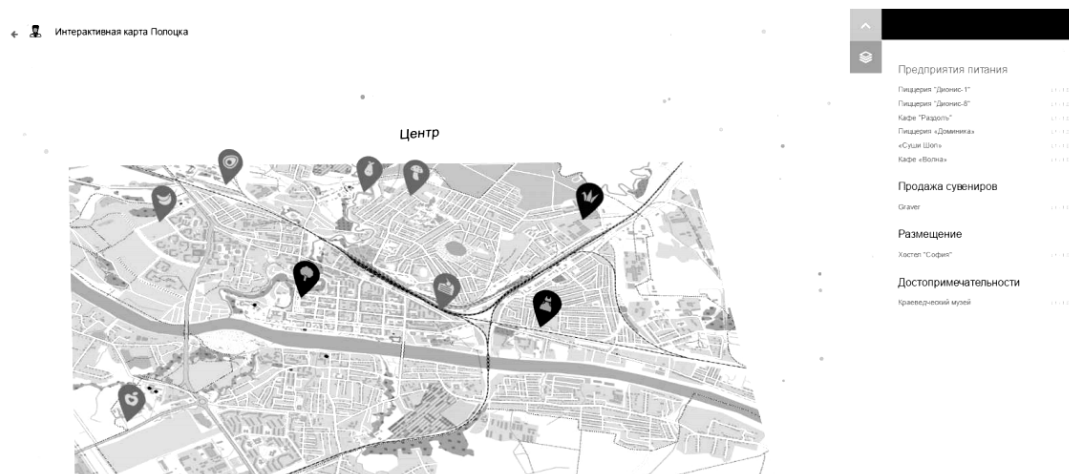


Рисунок 2. – Страница отдельного района Полоцка

На следующем шаге пользователю предоставляется блок с полной информацией о выбранном объекте. При нажатии кнопки «Смотреть полную схему общественного транспорта» выводится схема общественного транспорта Полоцка. Ссылка «Больше информации» перенаправляет пользователя на официальный сайт выбранного предприятия, если таковой имеется.

Остальные районы, представленные на сайте, имеют похожую структуру и выполняют те же функции. Переходы между районами можно осуществлять в любой момент работы сайта с помощью стрелок перехода или же перейдя ко всем районам и выбрав нужный. При нахождении и выборе объектов через поисковую форму осуществляется автоматический переход на нужный район города и вывод информации по необходимому объекту.

Разработанный программный продукт рассчитан на самую различную аудиторию, в частности туристические агентства, туроператоры, самостоятельные туристические группы, жители города, любознательные люди, желающие больше узнать о самом древнем городе Беларуси с целью упрощения процедуры поиска и составления туристических маршрутов через город Полоцк, также оказания помощи в изучении и сборе актуальной и полезной информации о главных объектах города.

Реализованный сайт является конкурентоспособным на рынке информационных технологий, так как имеет малую себестоимость, достаточный функционал и широкую информационную базу о главных туристических объектах, в отличие от существующих аналогов.

Разработанный путеводитель является легко расширяемым информационным ресурсом, что позиционирует его как проект с потенциалом развития в более крупную систему или интеграцию в уже существующую систему как самостоятельного модуля (например, как часть официального сайта города Полоцка).

Таким образом, можно сделать **вывод**, что на сегодняшний день технология WebGL, несмотря на ее недостатки, является самой простой для использования, производительной, оптимизированной и актуальной web-технологией для представления трехмерной графики в браузерах и Web-приложениях. Применение данной технологии предоставляет возможность разработки простого и удобного 3D-путеводителя, обладающего достаточной функциональностью для решения поставленных перед ним задач. Разработанный интерактивный путеводитель имеет удобную навигацию, большую базу главных объектов города, предоставляет систематизированную информацию о них, полностью адаптирован под русскоязычное население и является бесплатным программным продуктом.

ЛИТЕРАТУРА

1. WebGL – Интерфейсы веб API | MDN [Электронный ресурс]. – Режим доступа: http://www.developer.mozilla.org/ru/docs/Web/API/WebGL_API. – Дата доступа: 23.09.2017.
2. Введение в WebGL [Электронный ресурс]. – Режим доступа: <http://metanit.com/web/webgl/1.1.php>. – Дата доступа: 20.09.2017.
3. Разработка браузерной 3D-графики (графики для браузера), WebGL [Электронный ресурс]. – Режим доступа: <http://www.artofweb.ru/services/webgl-3d/>. – Дата доступа: 24.09.2017.

УДК 004:056.52

ВИДЫ АУТЕНТИФИКАЦИИ

О.И. РАЧИЦКИЙ*(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)*

Рассматриваются разнообразные типы и виды аутентификации, представленные в сфере безопасности веб-приложений. Отмечается наличие большого количества других протоколов и видов авторизации. Однако в данной работе речь пойдет о самых значимых, которые оказали влияние на индустрию и стали стандартами.

В целом процесс аутентификации представляет собой ответ на вопрос, «является ли клиент тем, за кого себя выдает». Проверки подлинности личности клиента осуществляются по самым различным протоколам и самыми различными способами.

В зависимости от степени доверительных отношений, структуры, особенностей сети и удаленности объекта проверка может быть односторонней или взаимной. Также различают однофакторную и строгую (криптографическую) аутентификации. Из однофакторных систем, наиболее распространенными на данный момент являются парольные системы аутентификации. У пользователя есть идентификатор и пароль, то есть секретная информация, известная только пользователю (и, возможно, системе), которая используется для прохождения аутентификации. В зависимости от реализации системы, пароль может быть одноразовым или многоразовым. В целом можно выделить следующие типы аутентификации: базовая, дайджест-аутентификация, с использованием средств протокола HTTPS, по предъявлению цифрового сертификата, с использованием куки; а также группа децентрализованных типов аутентификации: OAuth 1.0, OAuth 2.0, OpenAuth, OpenID Connect.

При использовании *базовой аутентификации* имя пользователя и пароль включаются в состав запроса. Любой злоумышленник, перехвативший пакет, способен получить доступ к секретной информации. Из ключевых недостатков такой аутентификации следует отметить невысокий уровень безопасности – пароль можно подсмотреть, угадать, подобрать, сообщить посторонним лицам, получить посредством перехвата сообщений между клиентом и аутентификационным сервером.

Далее рассмотрим *дайджест-аутентификацию*, при которой пароль пользователя передается в хешированном виде. Казалось бы, что по уровню безопасности передачи паролей этот тип мало чем отличается от базовой аутентификации, так как злоумышленнику предоставляется возможность перехватить сообщение в любом его виде, и злоумышленник все равно получает доступ к конечной точке. Но это не совсем так – пароль превращается в некоторый хеш по определенному алгоритму и всегда с добавлением произвольной строки символов, которая генерируется для каждого соединения. Таким образом, при каждом соединении генерируется новый хэш пароля и перехват его ничего не даст.

Протокол *HTTPS* позволяет шифровать все данные, передаваемые между браузером и сервером, а не только имена пользователей и пароли. Протокол HTTPS следует использовать в любом случае для обеспечения безопасности передаваемых данных. Это поддерживается практически всеми современными облачными платформами, включая выбранную для развертывания системы Heroku.

Механизмы аутентификации с применением *цифровых сертификатов*, как правило, используют протокол с запросом и ответом. Сервер аутентификации отправляет пользователю последовательность символов, так называемый запрос. В качестве ответа выступает запрос сервера аутентификации, подписанный с помощью закрытого ключа пользователя.

Множество различных сервисов и старых сайтов используют в качестве средства аутентификации *куки*. Если куки удастся похитить, то, подделав эти данные, можно аутентифицироваться в качестве другого пользователя. В случае, когда вводимые данные плохо фильтруются или не фильтруются вовсе, похитить куки становится очень просто. Для защиты куки от похищения методом запуска вредоносных JavaScript скриптов в браузере клиента используется флаг HTTPOnly, который поддерживают современные браузеры, однако это гарантирует защиту только внутри браузера, для защиты куки от перехвата во время передачи по незащищенным каналам используется флаг Secure, который шифрует куки средствами браузера. Тем не менее даже в этом случае куки все равно остаются уязвимостью с точки зрения безопасности аутентификации, так как браузер будет отправлять все куки с подходящим доменом при запросе на любой сайт из этого домена. На этой особенности построены CSRF (cross-request forgery) атаки.

Далее можно рассмотреть *децентрализованную аутентификацию*. Одним из главных минусов таких систем является то, что взлом дает доступ сразу ко многим сервисам. Однако взамен они предоставляют наивысший уровень безопасности, так как они изначально созданы и нацелены на осуществление

наибезопаснейшей аутентификации пользователей с созданием необходимых сессий и использованием самых последних разработок в области шифрования и безопасности.

OpenID – это открытая децентрализованная система аутентификации пользователей, она позволяет пользователю иметь одни аутентификационные данные для различных сервисов. Безопасность обеспечивается подписью всех сообщений. Возможной уязвимостью OpenID является подверженность атакам посредством перехвата запросов.

Многофакторная аутентификация используется для повышения безопасности посредством использования нескольких факторов аутентификации сразу. Но не всякая комбинация нескольких методов является многофакторной аутентификацией. Используются факторы различных типов: отличительные свойства (сканирование лица, отпечаток пальцев, радужная оболочка глаз); некоторое знание, которым обладает клиент (пароль, пин-код, последовательность действий); физическая вещь (карта доступа, пропуск, флеш-память, сим-карта с определенным телефонным номером).

Использование классических многоразовых паролей является серьезной уязвимостью при использовании открытых систем или незащищенных протоколов передачи данных. Это подтолкнуло разработчиков систем безопасности к созданию генераторов одноразовых паролей. Такие устройства или программы генерируют пароли для каждого соединения или сессии, чтобы при разрыве или перехвате сессии было невозможно подделать личность клиента. Для сохранения принципа двухфакторности аутентификации помимо сгенерированного устройством значения пользователь вводит постоянный пароль или подтверждает свой вход в систему вводом кода SMS-сообщения, которое на его личный номер высылает система.

Заключение

В результате проведенного исследования рассмотрены основные виды проверки подлинности клиентов на стороне приложений. Показано, что существует также много других видов аутентификации, более или менее популярных, каждый со своими плюсами и минусами. Очевидно, что правильное комбинирование систем аутентификации может значительно усложнить кражу данных клиентов.

ЛИТЕРАТУРА

1. Authentication Cheat Sheet [Электронный ресурс]. – Режим доступа: https://www.owasp.org/index.php/Authentication_Cheat_Sheet. – Дата обращения: 26.09.2017.
2. Understanding Authentication, Authorization, and Encryption [Электронный ресурс]. – Режим доступа: <https://www.bu.edu/tech/about/security-resources/bestpractice/auth/>. – Дата обращения: 26.09.2017.
3. Understanding and selecting authentication methods [Электронный ресурс]. – Режим доступа <http://www.techrepublic.com/article/understanding-and-selecting-authentication-methods/>. – Дата обращения: 26.09.2017.
4. Аутентификация и авторизация в микросервисных приложениях [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/company/dataart/blog/311376/>. – Дата обращения: 26.09.2017.

УДК 004:056.53

УЯЗВИМОСТИ В СОВРЕМЕННОМ КЛИЕНТ-СЕРВЕРНОМ ОБЩЕНИИ**О.И. РАЧИЦКИЙ***(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)*

Описаны основные виды атак на веб-приложения, которые используют уязвимости протоколов сообщения между клиентами и серверами, или недочеты архитектуры некоторых приложений, дизайн которых был создан без учета уязвимостей.

Современные браузеры являются достаточно защищенными от практически любых действий злоумышленников, ставящих своей целью похитить данные пользователей или воспользоваться пользовательскими полномочиями. Но в силу особенностей клиент-серверного взаимодействия структуры запросов между клиентом и сервером и некоторых особенностей JavaScript, используемого как язык работы с сайтом с клиентской стороны, существуют типы атак, которые невозможно предотвратить только силами браузера. О них и пойдет речь в этой работе.

Наиболее популярная и простая атака – это CSRF (cross-request forgery) атака. При CSRF атаке злоумышленники заставляют клиента выполнить нежелательные действия в приложениях, в которых клиент аутентифицирован. Этот тип атаки основан на недостатках протокола HTTP. Алгоритм действия злоумышленников в данном случае простой:

- найти на целевом сайте GET или POST запрос, выполнение которого клиентом может вызвать интерес со стороны злоумышленников. Это может быть запрос на денежный перевод или передачу личных данных;
- сформировать запрос таким образом, чтобы целевой сайт отослал данные или осуществил денежный перевод в том виде, в котором это выгодно мошенникам;
- создать собственный вредоносный сайт и поместить туда скрипт, который автоматически будет или осуществлять перенаправление на нужный сайт (для GET запросов), или активировать отправку формы с нужными параметрами (для POST запросов);
- дать ссылку в каком-либо виде пользователю или заставить пользователя перейти по ссылке, используя вредоносный код или скрипт, встроенный каким-либо образом в браузер клиента;
- как только пользователь перейдет на вредоносный сайт, на этот сайт не будут отправлены его куки, созданные на целевом сайте, однако как только вредоносный скрипт осуществит запрос к целевому сайту от лица пользователя, в таком запросе браузер автоматически передаст все куки домена целевого браузера, и если целью злоумышленников было проведение денежной транзакции, то целевой сайт проверит достоверность куки пользователя, а после авторизует и обработает сформированный злоумышленниками запрос. Это является основной уязвимостью при хранении аутентификационной и авторизационной информации в куки.

Вторым популярным типом атаки является XSS (Cross-Site Scripting). Суть XSS заключается в том, что разнообразными способами на страницу сайта, отображаемую у клиента, внедряется вредоносный код. Причем не всегда для этого требуется взлом самого сервера. XSS атаки можно разбить на два подтипа: отраженные и хранимые.

Отраженные XSS атаки основаны на отраженной уязвимости, на сегодняшний день является самой распространенной XSS-атакой. Эти уязвимости появляются, когда данные, предоставленные веб-клиентом, чаще всего в параметрах HTTP-запроса или в форме HTML исполняются непосредственно серверными скриптами для синтаксического анализа и отображения страницы результатов для этого клиента, без надлежащей обработки. Отраженная XSS-атака срабатывает, когда пользователь переходит по специально подготовленной ссылке, в которой в параметрах запроса передается вредоносный JavaScript код, не кодирующийся сервером и может быть исполнен в браузере клиента на странице, которую ему вернет сам сервер.

Отраженные атаки, как правило, рассылаются по электронной почте или размещаются на Web-странице. URL приманки не вызывает подозрения, указывая на надежный сайт, но содержит вектор XSS. Если доверенный сайт уязвим к вектору XSS, то переход по ссылке может привести к тому, что браузер жертвы начнет выполнять встроенный скрипт.

Хранимые XSS атаки являются наиболее разрушительными, потому что они возможны только тогда, когда злоумышленнику удастся внедрить на сервер вредоносный код, выполняющийся в браузере каждый раз при обращении к оригинальной странице. Классическим примером этой уязвимости являют-

ся форумы, блоги, чаты, на которых разрешено оставлять комментарии в HTML формате без ограничений. Также эта уязвимость имеет место быть, когда на сервере разрешается хранить пользовательские тексты и рисунки. Скрипты вставляются в эти тексты и рисунки.

Еще одной распространенной ранее, но возможной даже в современном мире веб-технологий, является *SQL-инъекция*. Внедрение SQL в зависимости от типа используемой СУБД и условий внедрения может дать возможность атакующему выполнить произвольный запрос к базе данных (например, прочитать содержимое любых таблиц, удалить, изменить или добавить данные), получить возможность чтения и/или записи локальных файлов и выполнения произвольных команд на атакуемом сервере.

Некоторые сервера опираются на клиентскую часть в плане составления запросов и не проверяют передаваемые параметры. Это может быть уязвимостью при внедрении в параметры SQL-кода, который будет исполнен сервером.

Также даже при защите от банального внедрения SQL-кода разработчики часто забывают о наличии оператора UNION в большинстве SQL СУБД, который призывает сервер выполнить помимо основной команды еще и дополнительную, описанную после этого оператора. Если говорить другими словами, язык SQL позволяет объединять результаты нескольких запросов при помощи оператора UNION. Это предоставляет злоумышленнику возможность получить несанкционированный доступ к данным.

Зачастую SQL-запрос, подверженный данной уязвимости, имеет структуру, усложняющую или препятствующую использованию UNION. В таких случаях злоумышленниками используется метод экранирования части запроса при помощи символов комментария в зависимости от типа СУБД. Для разделения команд в языке SQL используется символ *точка с запятой*, внедряя этот символ в запрос, злоумышленник получает возможность выполнить несколько команд в одном запросе, однако не все диалекты SQL поддерживают такую возможность.

Поиск скриптов, уязвимых для атаки, осуществляется злоумышленниками посредством изучения поведения скриптов сервера при манипуляции входными параметрами с целью обнаружения их аномального поведения. Манипуляция происходит всеми возможными параметрами: данными, передаваемыми через методы POST и GET, значениями куки, HTTP_REFERER (для скриптов), AUTH_USER и AUTH_PASSWORD (при использовании аутентификации). Как правило, манипуляция сводится к подстановке в параметры символа одинарной (реже двойной или обратной) кавычки.

Аномальным поведением считается любое поведение, при котором страницы, получаемые до и после подстановки кавычек, различаются (и при этом не выведена страница о неверном формате параметров). Наиболее частые примеры аномального поведения – выводится сообщение о различных ошибках, при запросе данных запрашиваемые данные не выводятся вообще, хотя страница отображается. Следует учитывать, что известны случаи, когда сообщения об ошибках в силу специфики разметки страницы не видны в браузере, хотя и присутствуют в ее HTML-коде.

Заключение

Несмотря на известность атак и общую осведомленность об их существовании избавиться от них невозможно из-за особенностей стандартов HTTP и модели серверов. Единственным вариантом защиты от такого типа атак является их предотвращение и контролирование необходимых для проведения атак условий на стороне сервера. Об этом следует помнить разработчикам сайтов.

ЛИТЕРАТУРА

1. Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), SQL Injection, HTML Injection, Etc [Электронный ресурс]. – Режим доступа: <https://www.zscaler.com/blogs/research/cross-site-scripting-xss-cross-site-request-forgery-csrf-sql-injection-html-injection-etc> . – Дата доступа: 26.09.2017.
2. CSRF and XSS: A Lethal Combination – Part I [Электронный ресурс]. – Режим доступа: <http://resources.infosecinstitute.com/csrf-xss-lethal-combination/>. – Дата доступа: 26.09.2017.
3. Understanding and selecting authentication methods [Электронный ресурс]. – Режим доступа: <http://www.techrepublic.com/article/understanding-and-selecting-authentication-methods/>. – Дата доступа: 26.09.2017.
4. XSS глазами злоумышленника [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/post/66057>. – Дата доступа: 26.09.2017.

УДК 004

**ЦЕЛЕСООБРАЗНОСТЬ ИСПОЛЬЗОВАНИЯ ТЕХНОЛОГИИ ASP.NET
И ЕЕ ПЛАТФОРМЫ WEBAPI ПРИ РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЙ**

С.Н. МАЛАШОНОК,
(Представлено: Ю.Н. КРАВЧЕНКО)

Рассматривается технология разработки ASP.NET, которая является частью платформы NetFramework и предназначена для написания клиент-серверных приложений и веб-сервисов, и платформы WebAPI, которая является частью ASP.NET. Анализируются целесообразность и преимущества ее использования.

ASP.NET представляет собой технологию, направленную на создание серверных веб-приложений и веб-сервисов различной сложности. Она содержит большое количество готовых элементов управления и различных технологий, начиная от технологии, предоставляющий доступ к базам данных и заканчивая технологией создания распределенных приложений.

Технология развивалась из более ранней версии технологии MicrosoftASP. Технология ASP имела ограничения по своим возможностям. Код программы вставлялся прямо в HTML-страницу в виде специальных тегов, что было неудобно и создавало путаницу.

Пример такой HTML-страницы представлен на рисунке 1.

```
<%@ Language=VBScript %>
<% Option Explicit%>
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" content="text/html">
<TITLE>
Hello ASP World!
<TITLE>
</HEAD>
<BODY>
<P><%
Dim i;
for i=1 to 5
Response.Write("<FONT SIZE=" & i)
Response.Write(">Этот код генерирует ASP!</FONT>")
next i
%></P>
</BODY>
</HTML>
```

Рисунок 1. – Пример файла ASP

Microsoft полностью перестроила ASP.NET, основываясь на CommonLanguageRuntime (CLR), которая является основой всех приложений Microsoft .NET. Разработчики могут писать код для ASP.NET, используя практически любые языки программирования, входящие в комплект .NET Framework (C#, Visual Basic.NET и JScript.NET) [1]. Как и все приложения .NET, приложения ASP.NET всегда компилируются. На первом этапе компиляции код компилируется в промежуточный язык IL, который на втором этапе компилируется в код на низкоуровневом машинном языке. IL-код создается только один раз и генерируется снова только при изменении исходного кода, а файлы, содержащие машинный код, кэшируются в системном каталоге. Благодаря этому механизму ASP.NET приложения не нуждаются в выполнении компиляции при каждом запросе веб-страницы.

В отличие от ASP, ASP.NET является объектно-ориентированной технологией, которая не только предоставляет коду доступ ко всем объектам .NETFramework, но и позволяет использовать все концепции объектно-ориентированного программирования. Технология позволяет создавать повторно используемые классы, использовать интерфейсы, расширять классы с помощью механизма наследования.

ASP.NET предоставляет возможность кэшировать ответы, сгенерированные страницей ASP.NET. Кэширование позволяет обработать запросы к одной и той же страницы с помощью кэша, при этом код, с помощью которого была создана страница, не будет выполняться повторно.

В ASP.NET3.5 появилась технология LanguageIntegratedQuery(LINQ), которая представляет собой набор расширений для языков C# и VisualBasic. Она позволяет писать на C# или VisualBasic код, способный манипулировать находящимися в памяти данными во многом в той же манере, что и запросы к базе данных. Самой главной разновидностью LINQ является LINQ to Objects, которая позволяет брать коллекцию объектов и выполнять запрос, извлекающий определенную информацию из некоторых этих объектов [2].

С выходом ASP.NET4.5 появился новый инструмент для создания веб-служб – WebAPI, который позволяет создавать HTTP-службы, доступные для широкого круга клиентов, включая браузеры и мобильные устройства.

Упрощение создания веб-служб является неотъемлемой особенностью Web API. Оно представляет собой значительное улучшение по сравнению с другими технологиями построения веб-служб, которые компания Microsoft предлагала на протяжении последнего десятилетия [3].

С помощью WebApi можно создать библиотеку классов, которая будет обрабатывать HTTP-запросы. Данная библиотека может быть расположена в исполняющей среде, благодаря чему к ней можно будет обращаться по HTTP. Открытые методы контроллеров являются конечными точками HTTP, которые могут возвращать данные в виде двоичных данных, JavaScriptObjectNotation (JSON), eXtensibleMarkupLanguage (XML) или обычного текста. Регулируемые правила маршрутизации позволяют задать форму UniformResourceLocator (URL), что значительно упрощает передачу данные в методы контроллера. Пример класса контроллера с методом представлен на рисунке 2.

```
public class ValuesController : ApiController
{
    // GET api/values
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }

    // GET api/values/5
    public string Get(int id)
    {
        return "value";
    }
}
```

Рисунок 2. – Метод контроллера

Таким образом, к основным преимуществам ASP.NET относятся:

- компиляция кода;
- использование объектно-ориентированного программирования;
- кэширование страниц;
- большое количество стандартных объектов, предоставляемых платформой .NetFramework;
- удобное создание веб-сервисов.

ЛИТЕРАТУРА

1. Принципы ASP.NET [Электронный ресурс]. – Режим доступа: https://studopedia.ru/2_30490_printsipi-ASPNET.html. – Дата доступа: 16.09.2017.
2. Мак-Дональд, М. MicrosoftASP.NET 4 с примерами на C# 2010 для профессионалов / Мэтью Мак-Дональд, Адам Фримен, Марио Шпуста. – 4-е изд. ; под ред. Ю.Н. Артеменко. – М. : И.Д. Вильямс, 2010. – 1424 с.
3. Одностраничные приложения [Электронный ресурс]. – Режим доступа https://professorweb.ru/my/ASP.NET/mvc/level8/8_1.php. – Дата доступа: 21.09.2017.

УДК 004.514

**РАЗРАБОТКА ИНТЕРФЕЙСА WEB-ПРИЛОЖЕНИЯ ДЛЯ ОПТИМИЗАЦИИ
ФОРМИРОВАНИЯ ГРАФИКОВ ОТПУСКОВ СОТРУДНИКОВ КОМПАНИИ****С.Н. МАЛАШОНОК,
(Представлено: Ю.Н. КРАВЧЕНКО)**

Рассматриваются способ создания пользовательского интерфейса web-приложения для оптимизации формирования графиков отпусков сотрудников компании. Интерфейс реализован с применением AngularJsFramework.

Управление отпусками работников – неотъемлемая часть менеджмента на предприятии. Наиболее ярко это выражается, когда предприятие работает с множеством проектов, над которыми трудится отдельная группа работников. Чтобы достичь максимальной производительности, необходимо четко следить за состоянием группы, так как отсутствие членов команды может негативно повлиять на весь рабочий процесс. Задача директора – составить и оптимизировать график отпусков работников, чтобы не понизить производительность работы предприятия.

Поскольку web-приложение предоставляет только часть функционала по управлению персоналом, то оно не требует большого количества web-страниц, поэтому было реализовано в виде SinglePageApplication (SPA).

SPA – это веб-приложение, размещенное на одной веб-странице, которая для обеспечения работы загружает весь необходимый код вместе с загрузкой самой страницы. Приложение такого типа появились сравнительно недавно, с началом эры HyperTextMarkupLanguage (HTML5) и SPA является типичным представителем приложений на HTML5.

HTML5 – это нечто иное, как HTML + CascadingStyleSheets (CSS3) + JavaScript и т.д. Таким образом, SPA – это приложения, написанные на языке JavaScript.

Первым плюсом стоит отметить тот факт, что приложения на SPA отлично работают на устройствах как стационарных, так и мобильных. «Большие» компьютеры, планшеты, смартфоны могут беспрепятственно работать с сайтами, построенных по принципу SPA.

Второй плюс – богатый пользовательский интерфейс. Так как веб-страница одна, построить богатый, насыщенный пользовательский интерфейс гораздо проще. Проще хранить информацию о сеансе, управлять состояниями представлений и управлять анимацией [1].

Для реализации пользовательского интерфейса использовался AngularJsFramework, который предназначен для разработки одностраничных приложений и использует шаблон Model-View-Controller (MVC).

Для описания интерфейса используется декларативное программирование, а бизнес-логика отделена от кода интерфейса, что позволяет улучшить тестируемость и расширяемость приложений.

Другой отличительной чертой фреймворка является двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом. Таким образом, AngularJS синхронизирует модель и представление.

Кроме того, AngularJs поддерживает такие функциональности, как AsynchronousJavascriptandXML (Ajax), управление структурой DocumentObjectModel(DOM), анимация, шаблоны, маршрутизация и т.д. Мощь фреймворка, наличие богатого функционала во многом повлияла на то, что он находит свое применение во все большем количестве веб-приложений [2].

На главной странице web-приложения отображается таблица с сотрудниками предприятия и количеством отработанных часов.

В верхней части страницы находятся фильтры календаря.

Поле ввода служит для фильтрации таблицы по проектам, фамилии и имени сотрудника.

Флаги используются для фильтрации календаря по типам отпусков.

Также можно построить график отпусков за любой период времени. Дни, на которые у сотрудника взят отпуск, отображаются разными цветами. По цвету можно будет определить статус заявки:

- красный – заявка не рассмотрена;
- зеленый – заявка подтверждена.

Если заявка сотрудника на отпуск накладывается на другую заявку или подтвержденный отпуск других сотрудников, работающих на совместных проектах, то заявка будет выделена специальным цветом и значком. При нажатии на значок график будет отсортирован по проектам сотрудника, который подал заявку на отпуск, и пересекающимся заявкам.

Страница графиков отпусков по месяцам представлена на рисунке 1.

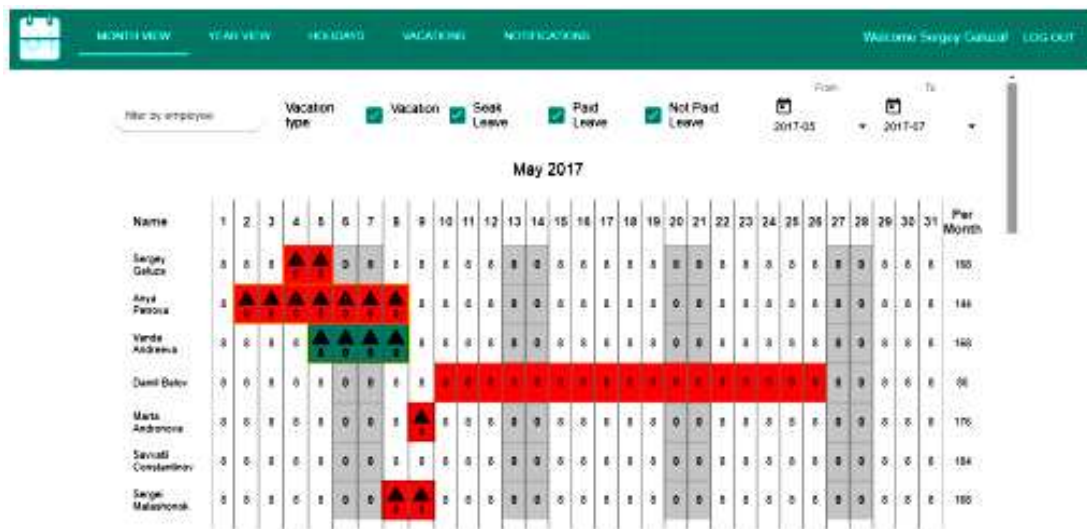


Рисунок 1. – Страница графиков отпусков по месяцам

Для учета и контроля личных отпусков разработана страница, представленная на рисунке 2.

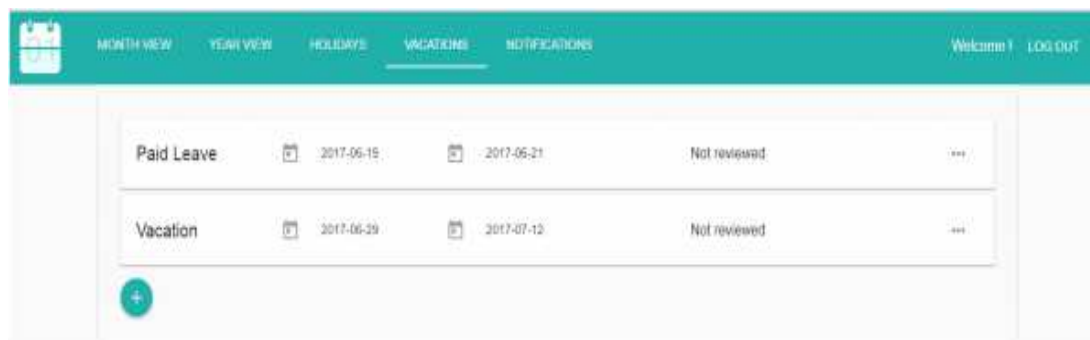


Рисунок 2. – Страница просмотра заявок на различные виды отпусков

В блоке отпуска отображаются тип отпуска, дата начала и конца, статус. Также на этой странице предусмотрена возможность создавать заявки на отпуска, после чего администратору будет выслано уведомление о том, что была создана новая заявка, а статус отпуска установится как непросмотренный.

В разработанном web-приложении реализованы функции добавления, редактирования и удаления заявок и отпусков работников предприятия, управления графиками отпусков, установки сокращенных и праздничных дней, установки количества отработанных часов сотрудниками, переноса рабочих дней, а также реализован полноценный сервис уведомлений.

Данное web-приложение должно упростить процессы формирования графиков отпусков и подачи сотрудниками заявок на отпуска.

ЛИТЕРАТУРА

1. Что такое SPA или одностраничный портал [Электронный ресурс]. – Режим доступа: <http://www.codenet.ru/webmast/js/spa/>. – Дата доступа: 18.09.2017.
2. Введение в AngularJS [Электронный ресурс] – Режим доступа: <https://metanit.com/web/angular/1.1.php>. – Дата доступа: 20.09.2017.

УДК 004.056.55

**ПРОЕКТИРОВАНИЕ СТЕГОСИСТЕМЫ,
ОСНОВАННОЙ НА СОКРЫТИИ ТЕКСТОВЫХ ДАННЫХ
В АУДИОФАЙЛАХ ЗА СЧЕТ ИЗМЕНЕНИЯ ВРЕМЕНИ ЗАДЕРЖКИ ЭХО-СИГНАЛА**

А.В. КОХАНОВСКИЙ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматривается практический способ создания надежной стегосистемы с нестандартным способом скрытия информации. Построена система, основанная на скрытии текстовых сообщений в аудиофайлах. Выяснено, насколько пригодна такая система для практического применения.

В статье пойдет речь о сокрытии данных в аудиофайлах с помощью одного из методов стеганографии. Под цифровой стеганографией понимается скрытие одной информации в другой. Причем сокрытие это должно реализовываться таким образом, чтобы, во-первых, не были утрачены свойства и некоторая ценность скрываемой информации, а во-вторых, неизбежная модификация информационного носителя, не только не уничтожила смысловые функции, но и на определенном уровне абстракции даже не меняла их. Тем самым факт передачи одного сообщения внутри другого не выявляется традиционными методами.

Принцип построения стеганографической системы и описание алгоритма скрытия данных. Стандартная стеганографическая схема сохраняется вне зависимости от технологии, которой она реализуется. Задачу встраивания и выделения сообщений из другой информации выполняет стегосистема. Стегосистема состоит из следующих основных элементов, представленных на рисунке 1.

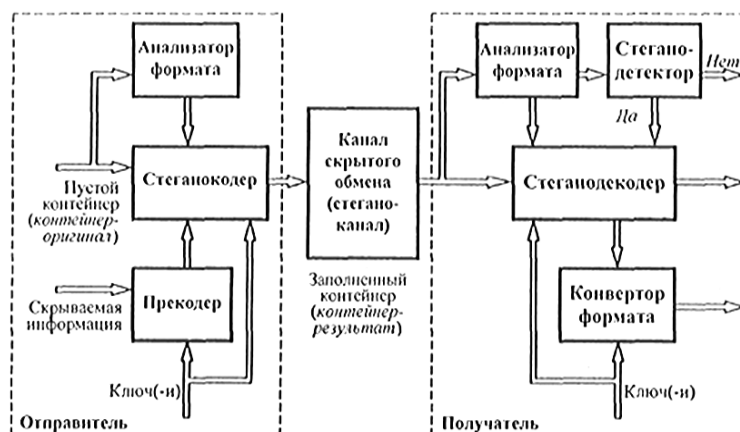


Рисунок 1. – Структурная схема типичной стегосистемы

Контейнер – любая информация, предназначенная для сокрытия тайных сообщений. *Стегоключ*, или просто *ключ*, – секретный ключ, необходимый для сокрытия информации. *Прекодер* – устройство, предназначенное для преобразования скрываемого сообщения к виду, удобному для встраивания в сигнал-контейнер. *Стегокодер* – устройство, предназначенное для осуществления вложения скрытого сообщения в другие данные с учетом их модели; устройство выделения встроенного сообщения. *Стегодетектор* – устройство, предназначенное для определения наличия стегосообщения. *Декодер* – устройство, восстанавливающее скрытое сообщение. Этот узел может отсутствовать.

Данные, содержащие скрытое сообщение, могут подвергаться преднамеренным атакам или случайным помехам. Как показано на рисунке 1, в стегосистеме происходит объединение двух типов информации так, чтобы они могли быть различимы двумя принципиально разными детекторами. В качестве одного из детекторов выступает система выделения цифровых водяных знаков (ЦВЗ), в качестве другого – человек [1].

Для сокрытия данных использовался метод встраивания информации за счет изменения времени задержки эхо-сигнала. Этот метод позволяет внедрять данные в сигнал прикрытия, изменяя параметры эхо-сигнала. К параметрам эхо, несущим внедряемую информацию, относятся: начальная амплитуда, время спада и сдвиг (время задержки между исходным сигналом и его эхо). При уменьшении сдвига два сигнала смешиваются. В определенной точке человеческое ухо перестает различать два сигнала, и эхо воспринимается, как добавочный резонанс. Кодер использует два времени задержки: одно для кодирования нуля, другое для кодирования единицы. И то, и другое время задержки меньше того, на котором человеческое ухо может распознать эхо. Кроме уменьшения времени задержки необходимо добиться уста-

новлением начальной амплитуды и времени спада того, чтобы внедренная информация не могла быть воспринята системой слуха человека.

При кодировании информации были выбраны значения задержек эхо-сигналов, равные 0,0012 и 0,0008 секунды. Данные значения снижают эффективность алгоритма извлечения, уменьшая вероятность правильного извлечения бита, однако подходит практически для всех типов контейнеров. Значение амплитуды, равное 40%, выбрано в связи с тем, что оно помогает увеличить вероятность того, что факт наличия скрытой информации не будет раскрыт.

Декодирование внедренной информации представляет собой определение промежутка времени между сигналом и эхо. Для этого необходимо рассмотреть амплитуду (в двух точках) автокорреляционной функции дискретного косинусного преобразования логарифма спектра мощности (кепстра). Всплеск автокорреляционной функции будет наблюдаться через δ_1 или δ_0 секунд после исходного сигнала (рис. 2). Правило декодирования основано на определении промежутка времени между исходным сигналом и всплеском автокорреляции [2].

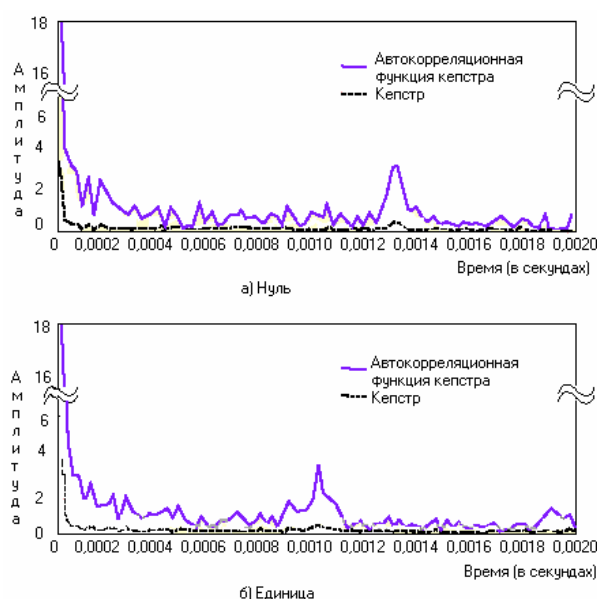


Рисунок 2. – Поведение автокорреляционной функции

По исследованиям В. Бендера и Н. Моримото, данная схема позволяет внедрять 16 бит в одну секунду аудиозаписи незаметно, без потери ее качества [3]. Выбор метода встраивания информации в звуковые файлы, используя преобразования эхо-сигналов, обусловлен тем, что этот метод имеет устойчивость к амплитудным и частотным атакам, что позволяет обойти остальные, неустойчивые к этим атакам, методы. Однако данный метод неустойчив к временным атакам, напрямую влияющим на длину аудио-файла и количество отсчетов в нем. При реализации метода стеганографической защиты информации посредством преобразования эхо-сигналов стало ясно, что основную трудность представляет реализация наиболее эффективного алгоритма извлечения встроенных бит. Некоторые исследования в этом направлении показали, что возможно достичь наибольшей эффективности только посредством индивидуального подхода к каждому контейнеру отдельно, меняя при этом время задержек накладываемого эхо-сигнала.

Заключение. Исследования в области стеганографии – перспективное направление защиты информации, так как в современном мире задача передачи секретной информации стоит наравне со скрытым общением, т.е. скрытия факта передачи сообщений. Поэтому необходимо продолжать исследованиями в этой области для поиска новых, эффективных, методов или улучшения существующих. В данной работе был рассмотрен метод встраивания информации в звуковые файлы. Метод эхо-сигналов – самый перспективный, однако требует доработки в плане пропускной способности, и вероятности правильного извлечения встроенных бит информации. Программный продукт реализован и готов к использованию с возможностью доработки.

ЛИТЕРАТУРА

1. Matsui. K. Digital signature on a facsimile document by recursive MH coding / K. Matsui, K. Tanaka and Y. Nakamura // Symposium On Cryptography and Information Security, 1989.
2. Садов, В.С. Компьютерная стеганография / В.С. Садов. – М. : МГВРК, 2012. – 289 с.
3. Грибунин, В.Г. Цифровая стеганография / В.Г. Грибунин. – М. : СОЛОН-Пресс, 2002. – 272 с.

УДК 004.056.55

**ОПРЕДЕЛЕНИЕ СТЕПЕНИ ПРИГОДНОСТИ КОНТЕЙНЕРА
ДЛЯ СТЕГАНОГРАФИЧЕСКОЙ МОДИФИКАЦИИ В СТЕГОСИСТЕМЕ,
ОСНОВАННОЙ НА СОКРЫТИИ ТЕКСТОВЫХ ДАННЫХ В АУДИОФАЙЛАХ
ЗА СЧЕТ ИЗМЕНЕНИЯ ВРЕМЕНИ ЗАДЕРЖКИ ЭХО-СИГНАЛА**

А.В. КОХАНОВСКИЙ

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Определяется степень пригодности контейнера для стеганографической модификации. Представлены некоторые виды атак на стегосистему. Построена система, основанная на скрытии текстовых сообщений в аудиофайлах, а также изучены атаки и выяснено, насколько пригодна такая система для практического применения.

Основными параметрами аудиосигналов являются их амплитуда, частота и фаза. Для стеганографической модификации пригодны все эти параметры, но перед процедурой встраивания необходимо оценить пределы модификации параметров того или иного контейнера, а также вносимые при этом искажения. Пропускная способность, надежность, устойчивость стеганографической системы во многом будет определяться степенью модификации аудиоконтейнера [1].

Оценка стеганостойкости реализованной системы

Существует множество классификаций оцифрованных аудиосигналов по их техническим параметрам, таким как, частота дискретизации, количество каналов (моно или стерео), качество записи (количество кб в секунду) и т.д. В то же время, если аудиосигналы (контейнеры) принадлежат к одному формату представления данных (например, WAVE) имеют одну частоту дискретизации, одинаковое количество каналов, схожее качество звука и различаются только по музыкальному жанру, достаточно сложно оценить пригодность того или иного контейнера к стеганографической модификации.

Все аудиосигналы можно разбить на три группы (рис. 1):

- аудиофайлы, подобные речи, с большим количеством пауз: стихи, голос диктора, речь с негромкой фоновой музыкой и т.д. (рис. 1, а);
- аудиофайлы, не обладающие широким частотным диапазоном, подобные обычным музыкальным композициям: эстрадные песни, популярная музыка, блюз и т.д. (рис. 1, б);
- аудиофайлы, обладающие широким частотным диапазоном, прошедшие дополнительную компьютерную обработку: электронная музыка (trance, house, techno и др.), синтезированные звуки природы и т.д. (рис. 1, в).

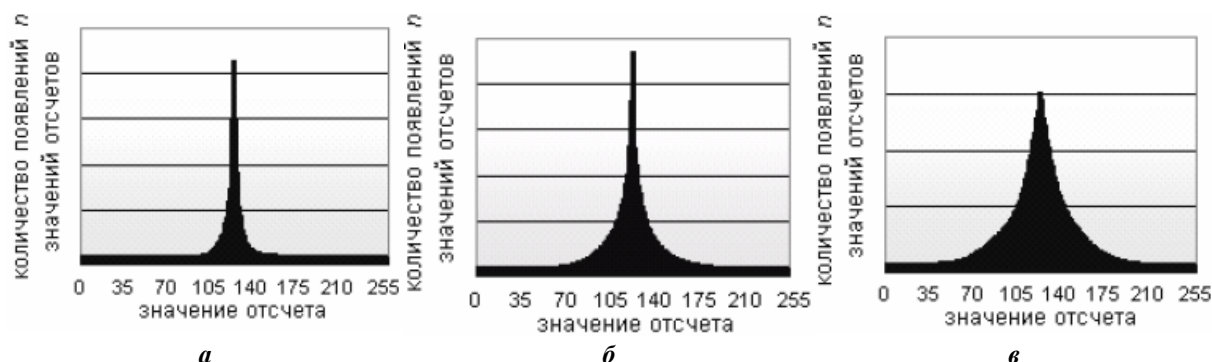


Рисунок 1. – Графики распределения отсчетов для трех групп аудиосигналов

Использование аудиофайлов из первой группы оказалось совсем неприемлемым для использования независимо от качества использованного контейнера. Вероятность правильного и полного изъятия сообщения из контейнера достаточно мала. При использовании аудиофайлов из второй группы, как и ожидалось, искажения едва различимы натренированным ухом, а вероятность правильного изъятия достаточно велика. Использование аудиофайлов из третьей группы показало, что вероятность правильного извлечения близка к 100%, при этом различить пустой контейнер и контейнер с данными на слух не удастся.

Из этого можно сделать вывод, что лучше всего для скрытия данных методом задержки эхо-сигнала подходят аудиофайлы, прошедшие дополнительную компьютерную обработку (электронная му-

зыка). Также следует учесть частоту дискретизации и количество каналов: чем их значения выше, тем больше вероятность правильного изъятия [2].

Основные виды атак на стегосистемы:

1. *Атака на основе известного заполненного контейнера.* Задача состоит в обнаружении факта наличия скрытого сообщения. Прослушивание аудиосигнала с целью обнаружить какие-либо помехи. Был выбран контейнер длительностью 60 секунд, с динамичной музыкой. И произведено скрытие сообщения длиной 20 символов. Результат – прослушивание заполненного контейнера на акустической системе показало, что никаких посторонних звуков, хрипов и тресков не обнаружено.

2. *Атака на основе известного пустого контейнера.* Если он известен, то путем сравнения его с предполагаемым контейнером, в котором есть скрытое сообщение, всегда можно установить факт наличия стегоканала. Это может быть сравнение по длине аудиосигнала, сравнение по размеру, а также прослушивание. Был выбран контейнер длительностью 60 секунд, с динамичной музыкой. И произведено скрытие сообщения длиной 20 символов. Результат – многократное прослушивание заполненного контейнера и контейнера оригинала на акустической системе без проигрывания низких частот (ноутбук) показало, что отличий между звучанием нет. Однако при прослушивании на акустической системе с устройством, проигрываемым низкие частоты, стали заметны некоторые отличия в глубине звучания заполненного контейнера и контейнера оригинала.

3. *Атаки против встроенного сообщения,* направленные на удаление или порчу встроенной информации путем манипулирования заполненным контейнером, а также направленные на затруднение или невозможность правильной работы детектора. При этом сообщение в аудиофайле остается, но теряется возможность его приема. Примерами являются: обрезка (изменение длины) аудиосигнала; изменение формата на другой; изменение параметров формата контейнера, ускорение или замедление; изменение тональности. Результат – метод скрытия данных с помощью задержки эхо-сигнала неустойчив к временным изменениям. При изменении длины аудиофайла на несколько секунд извлечь какую-либо информацию уже невозможно. Изменение любых других параметров аудиофайла не повлияло на получение скрытой информации [3].

Заключение

Рассмотрен метод встраивания информации в звуковые файлы. Метод эхо-сигналов – самый перспективный, однако требует доработки в плане пропускной способности и вероятности правильного извлечения встроенных бит информации. Для оценки эффективности рассматривались такие параметры, как особые требования к контейнерам – звуковым файлам и сложность определения ее злоумышленником, влияние попытки уничтожения скрытой информации на сохранность контейнера. Выбор метода встраивания информации в звуковые файлы, используя преобразования эхо-сигналов, обусловлен тем, что этот метод имеет устойчивость к амплитудным и частотным атакам, что позволяет обойти остальные, неустойчивые к этим атакам, методы. Однако данный метод неустойчив к временным атакам, напрямую влияющим на длину аудиофайла и количество отсчетов в нем.

Программа реализована и готова к использованию с возможностью ее доработки.

ЛИТЕРАТУРА

1. Садов, В.С. Компьютерная стеганография / В.С. Садов. – М. : МГВПК, 2012. – 289 с.
2. Matsui, K. Digital signature on a facsimile document by recursive MH coding / K. Matsui, K. Tanaka and Y. Nakamura // Symposium On Cryptography and Information Security, 1989.
3. Грибунин, В.Г. Цифровая стеганография / В.Г. Грибунин. – М. : СОЛОН-Пресс, 2002. – 272 с.

УДК 004.021

LibGDX – КРОССПЛАТФОРМЕННЫЙ ФРЕЙМВОРК**Р.А. КАРЦЕВИЧ***(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматривается кроссплатформенный фреймворк, применяемый в создании различного рода приложений. Показано, что кроссплатформенный фреймворк LibGDX является отличным инструментом в создании игровых приложений благодаря своей простоте и использованию таких языков, как C и C++, для обеспечения более быстрой работы.

В основном создание игровых приложений подразумевает за собой использование какого-либо фреймворка. *Фреймворки* – это программные продукты, которые упрощают создание и поддержку технически сложных или нагруженных проектов. Фреймворк, как правило, содержит только базовые программные модули, а все специфичные для проекта компоненты реализуются разработчиком на их основе.

Плюсы фреймворков:

- разработка на фреймворке (в отличие от самописных решений) позволяет добиться простоты сопровождения проекта;
- возможна (и относительно проста) реализация любых бизнес-процессов, а не только тех, которые изначально заложены в систему. Также проекты на базе фреймворков легко масштабируемы и модернизируемы;
- решения на фреймворках, как правило, работают значительно быстрее и выдерживают большую нагрузку, чем CMS и самописные системы. Именно поэтому много популярных интернет-магазинов работают не на коробочных CMS, а на фреймворках. По уровню безопасности решения на фреймворках значительно превосходят самописные системы и сравнимы с CMS (как правило, сайты на фреймворках даже безопаснее).

Минусы фреймворков:

- сроки разработки типового функционала на фреймворках больше, чем при использовании CMS. Фреймворки содержат только базовые компоненты бизнес-логики уровня приложения, поэтому многие функции реализовываются индивидуально;
- для разработки на фреймворке требуется понимание бизнес-процессов, которые требуется реализовать. Например, если в CMS уже есть некий предустановленный процесс обработки заказов, то фреймворки такого не предоставляют;
- одним из достойных кандидатов является фреймворк LibGDX, благодаря которому облегчается создание кроссплатформенных приложений.

LibGDX – кроссплатформенный фреймворк для разработки игр и визуализации, основанный на языке программирования Java с некоторыми компонентами, написанными на C и C++ для повышения производительности определенного кода. В настоящее время поддерживает Windows, Linux, Mac OS X, Android, iOS и HTML5 как целевые платформы.

Одними из выдающихся возможностей фреймворка являются:

- поддержка Windows, Linux, Mac OS X, Android ОС, Браузеры с поддержкой WebGL и iOS.32 и 64-разрядные версии ОС;
- гибкость и увеличение возможностей достигается за счет расширений. Можно подключить физический движок Box2D для работы с объектами и физикой реального мира, добавить поддержку TrueType шрифтов или работать с 3D-объектами;
- встроенный набор виджетов для рисования графического интерфейса – как игр, так и программ;
- наличие возможности легко реализовать логику поведения частиц (к примеру, можно сделать красивый взрыв);
- высокая производительность.

LibGDX позволяет написать код однажды и затем развертывать игру или приложение на нескольких платформах без модификации. Также в нем присутствует такая возможность, как разрабатывать приложение на основном компьютере и получать огромную выгоду быстрой разработки, вместо того чтобы ждать, когда последние изменения будут внедрены и установлены на устройство и будут скомпилированы в HTML5. Можно воспользоваться всеми инструментами языка программирования Java, для увеличения роста продуктивности, насколько это возможно.

LibGDX позволяет перейти на такой низкий уровень, какой требуется в той или иной ситуации, давая прямой доступ к файловой системе, устройствам ввода, аудио устройствам и OpenGL через единый OpenGL ES 2.0 и 3.0 интерфейс.

Наверху таких низкоуровневых возможностей создан мощный набор API, который позволяет решать общие в разработке игр задачи, такие как визуализация спрайтов, теста, построение пользовательских интерфейсов, проигрывание звуковых эффектов и музыки, линейная алгебра и тригонометрические вычисления, разбор JSON и XML, и так далее.

При необходимости, libGDX может перейти от Java к нативному коду, чтобы сосредоточиться на самой лучшей и возможной производительности. Весь этот функционал скрывается за Java API, так что можно не беспокоиться о кросс-компилировании нативного кода на всех платформах. Многие части libGDX знают специфику платформы и не нужно с ними сталкиваться.

LibGDX сосредоточен на том, чтобы являться фреймворком, нежели двигателем, признавая, что нет универсального решения. LibGDX предоставляет мощные абстракции, которые позволяют выбирать, как создавать игру или приложение.

Заключение

Кроссплатформенный Фреймворк LibGDX является отличным инструментом в создании игровых приложений благодаря своей простоте и использованию таких языков, как C и C++, для обеспечения более быстрой работы.

ЛИТЕРАТУРА

1. LibGDX // Википедия : свободная энцикл. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/LibGDX>. – Дата доступа: 25.09.2017.
2. Фреймворки // Webcreator – интернет-портал [Электронный ресурс]. – Режим доступа: https://webcreator.ru/articles/about_frameworks. – Дата доступа: 26.09.2017.
3. LibGDX – фреймворк для разработки игр // Libgdx.ru – интернет-портал [Электронный ресурс]. – Режим доступа: <http://www.libgdx.ru/2013/08/introduction.html#more>. – Дата доступа: 26.09.2017.

УДК 004.928

СОЗДАНИЕ 2D-АНИМАЦИИ ОБЪЕКТОВ

Р.А. КАРЦЕВИЧ

(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Рассматривается подход к созданию 2D-анимации объектов в игровых проектах. Показан один из способов создания анимации – способ раскадровки специально подготовленного изображения. Анимация создается за счет воспроизведения на экране кадров с определенной частотой.

Анимация играет существенную роль в повседневной жизни человека. Сегодня анимация применяется в самых различных сферах деятельности, одними из них являются:

- создание спецэффектов в кинематографе;
- создание рекламы;
- разработка компьютерных игр;
- создание анимационных фильмов;
- разработка интерфейсов сайтов и web-приложений;

Наибольшую популярность анимация получила в компьютерных играх, которая «оживляет» игру, делая ее более красочной и привлекательной, что в свою очередь привлекло огромное внимание большого количества игроков.

2D-анимация – техника, используемая для создания иллюзии движения, используя статические изображения. Анимация состоит из множества кадров, которые показываются в последовательности через определенные интервалы времени.

Рассмотрим процесс создания анимации на примере горящего огня. Анимацию горящего пламени можно достичь путем использования изображений во время полыхания пламени и бесконечным проигрыванием этих изображений в обратном порядке. На рисунке 1 показан полный цикл горения.

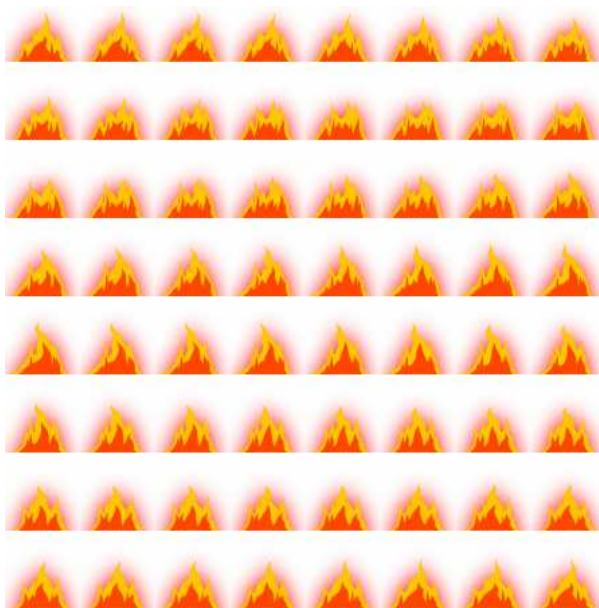


Рисунок 1. – Полный цикл горения пламени

Такое изображение называется *спрайт-листом*. Каждая область является спрайтом и называется кадром. Для создания анимации горения, спрайты должны быть нарисованы один за другим, по истечении определенного времени. В зависимости от того, как быстро должно гореть пламя, нужно определить, сколько времени кадр будет оставаться на экране. *Кадровая частота* – количество сменяемых кадров за секунду. Посмотрев на спрайт-лист, мы увидим полный цикл горения, состоящий из 64 кадров. Если полный цикл бега персонажа укладывается в одну секунду, то мы должны показывать 64 кадра в секунду. Это дает нам кадровую частоту в 64 FPS (FramesPerSecond). Двигаясь дальше, нетрудно рассчитать время состояния (время кадра), которое показывает, сколько времени кадр должен отображаться на экране перед тем, как его сменит следующий кадр: $1 \text{ секунда} / 64 = 0,064$. Другими словами, чтобы

анимация была в 64 FPS, кадр должен сменяться каждые 0,064 секунды. Анимация является простым конечным автоматом. Так, горящее пламя имеет 64 состояния, согласно спрайт-листу, представленному на рисунке 2. Пронумерованные кадры горения представляют состояние, через которое оно проходит.

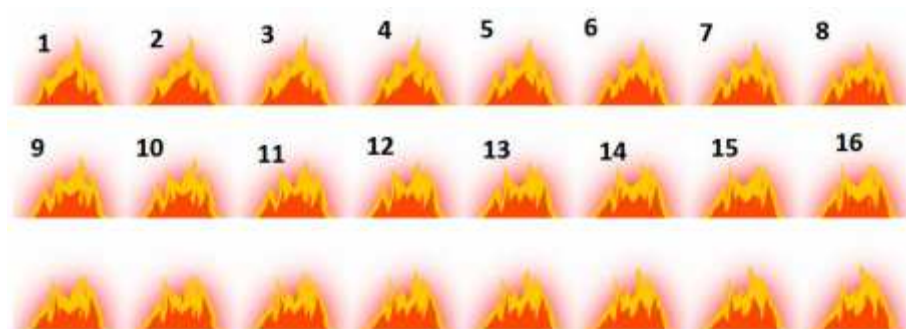


Рисунок 2. – Последовательность формирования кадров анимации

В любой момент времени конечный автомат может находиться в двух или более состояниях. Когда пламя находится в состоянии 1, то рисуется спрайт, связанный с этим состоянием. Пламя находится в этом состоянии 0,064 секунды и как только это время проходит, горение переходит в следующее состояние, которое равно 2. Это продолжается до тех пор пока не будет достигнуто последнее состояние или кадр. Циклическая анимация означает переход на начало, когда анимация достигает последнего кадра. Использовать анимацию в libGDX чрезвычайно просто. Есть одно ограничение, касающееся размера спрайт-листа, которое нужно запомнить: когда используется OpenGL 1.x, размер спрайта должен быть степенью двойки.

Заключение

В результате проведенного исследования выяснили, что одним из способов создания анимации является способ раскадровки специально подготовленного изображения. После чего с определенной частотой кадры воспроизводятся на экране, за счет этого и создается анимация.

ЛИТЕРАТУРА

1. Компьютерная анимация // Википедия – свободная энцикл. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Компьютерная_анимация. – Дата доступа: 25.09.2017.
2. Создание эффекта движения с помощью спрайтов // VirtualDub – Интернет-портал [Электронный курс]. – Режим доступа: <http://virtualdub.epublish.ru/p33aa1.html>. – Дата доступа: 26.09.2017.

УДК 004.457

**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОНТРОЛЯ ПОСЕЩАЕМОСТИ СТУДЕНТОВ
НА ОСНОВЕ АНАЛИЗА ОТПЕЧАТКОВ ПАЛЬЦЕВ****К.А. АПАНЧЁНОК****(Представлено: М.В. МАТЮШ)**

Проанализирована работа приложения для компьютера системы контроля посещаемости студентов с использованием отпечатков пальцев. Программная часть разработана на языке программирования Java для ОС Windows.

Разработанное приложение в рамках данной работы осуществляет функции, позволяющие работать с устройством и удаленной базой данных. Приложение связывает собой два этих компонента при работе с пользователем посредством отправки SQL команд на сервер БД [1] и передачи команд на устройство через COM-порт [2]. К основным функциям приложения можно приписать функции, такие как чтение/запись данных в удаленную базу данных, отправка и прием команд с устройства и вывод информации для пользователя. Все эти функции взаимодействуют между собой при работе с программой. Для более подробного анализа данных функций рассмотрим работу программы в целом.

Программа при запуске представляет собой окно с возможностью выбора основных параметров, просмотром состояния соединения сканера и базы данных. В нем по умолчанию отображена пустая таблица посещаемости, неактивные вкладки кнопок управления таблицей и меню. Меню состоит из двух пунктов: «Файл» (для авторизации и открытия базы данных) и «Параметры» (для редактирования БД администратором).

Пользователем программы является преподаватель. Для начала работы он должен открыть базу данных определенного предмета и группы. За это отвечает окно выбора основных параметров (факультет, кафедра, преподаватель, предмет, группа). При выборе параметров следующие значения сразу подгружаются в выпадающий список, исключая возможность неверного выбора с ошибкой запроса к базе данных (рис. 1).

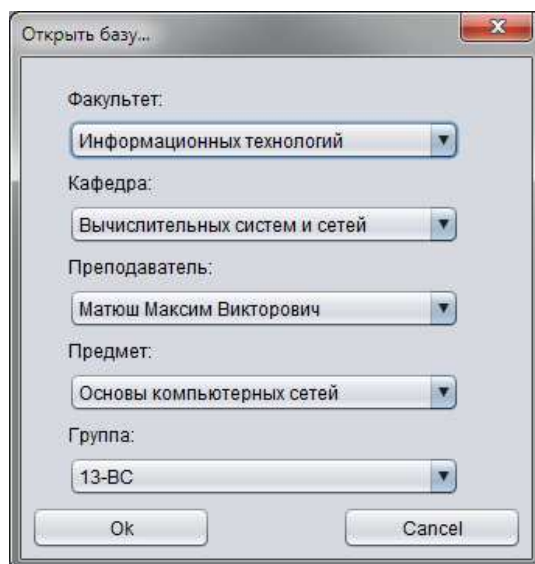


**Рисунок 1. – Окно выбора основных параметров
для начала работы**

После выбора преподавателя система проверяет на наличие у него прав администратора. Данная информация содержится в БД для каждого преподавателя (если администратор, то значение поля равно «1», если нет – «0»). Если таковы имеются, то пользователь получает возможность прекратить выбор предмета и группы для ведения журнала, а сразу же продолжить работу в режиме администратора. Иначе, после выбора всех параметров и подтверждения появляется диалоговое окно с двумя кнопками, которые уточняют тип работы с программой: режим редактирования базы данных или же режим ведения журнала. После выбора режима редактирования базы пользователю открывается доступ к пункту меню «Параметры»,

в котором собраны функции, необходимые для добавления, изменения или удаления записей в базе данных. При выборе режима ведения журнала в базу данных делается SQL запрос на получение списка студентов, списка отпечатков пальцев и списка посещения на выбранный день (если не выбран, берется текущий). Список отпечатков отправляется на устройство с порядковыми номерами студентов. В таблицу главного окна загружается список студентов выбранной группы, активируется вкладка управления посещаемостью с набором кнопок для сканирования, отметки и отправки данных на сервер (рис. 2).

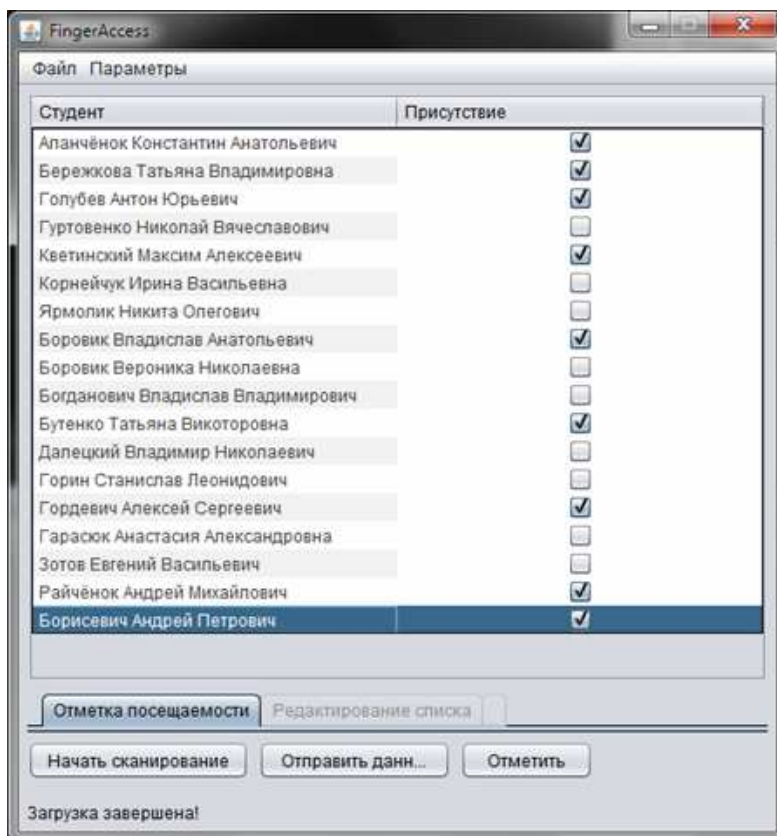


Рисунок 2. – Окно программы в режиме ведения журнала

Далее пользователь может отметить присутствующих студентов путем нажатия на кнопку «Начать сканирование» и/или «Снять/поставить присутствие». После нажатия на кнопку сканирования на устройство отправляется команда на скан и поиск отпечатка в БД сканера; кнопка меняет название на «Завершить сканирование». После этого устройство отправляет данные о нахождении совпадения в БД или нет. Если совпадение было найдено, то в таблице напротив совпадающего студента появится галочка, если нет – устройство продолжит работу. Сканирование завершается тогда, когда все студенты были отмечены, или же по нажатию пользователем кнопки «Завершить сканирование». Далее преподаватель может либо отредактировать данные вручную (если это необходимо), либо отправить данные на сервер (т.е. сохранить) нажав на соответствующую кнопку.

Преподаватель также может просмотреть посещаемость по предмету для группы за указанную дату, которая имеется в журнале. Для этого ему необходимо оторвать окно данной функции в меню и ввести значения параметров (факультет, кафедра, группа, предмет, дата). Далее в таблицу загружается список студентов и их посещаемость в выбранную дату. При необходимости, посещаемость может быть отредактирована преподавателем-администратором.

Администратор имеет расширенные возможности для управления базой данных, такие как добавление/редактирование/удаление групп, предметов, редактирование списков предметов групп. В режиме редактирования группы в таблицу загружается список студентов и шаблоны отпечатков пальцев соответственно из БД отправкой SQL запроса. Также активируется вкладка с набором функций для редактирования списка студентов. Она состоит из трех кнопок: «Добавить», «Редактировать» и «Удалить».

При добавлении студента появляется окно с двумя полями, отвечающими за ввод фамилии, имени, отчества и шаблона отпечатка пальца соответственно (рис. 3).

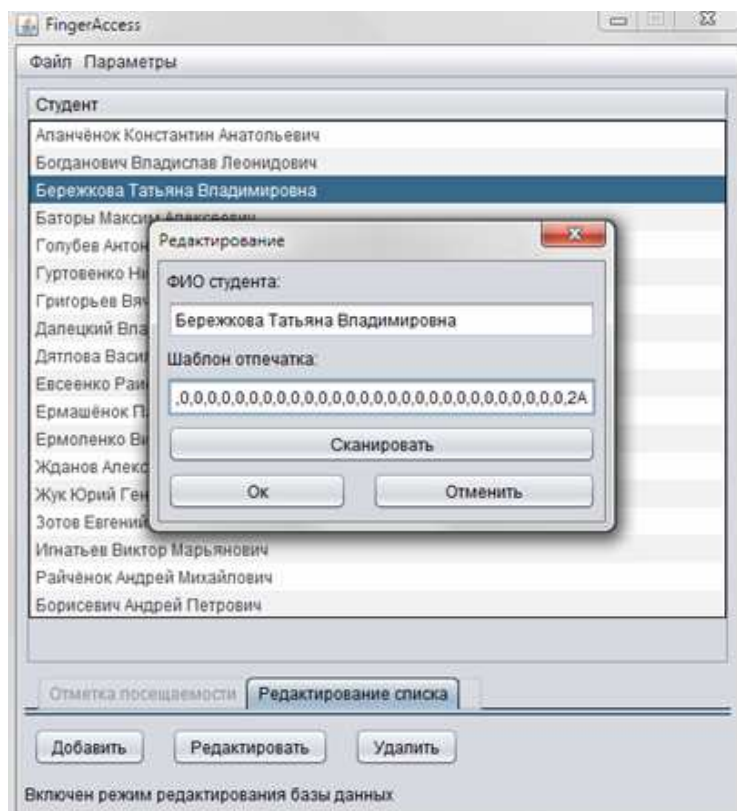


Рисунок 3. – Окно программы в режиме редактирования группы и добавления/редактирования студента

Для ввода шаблона отпечатка необходимо нажать кнопку «Сканировать». Далее программа отправляет на устройство команду сканирования пальца и отправки готового шаблона. Программа принимает данный шаблон и записывает в соответствующее поле. При сохранении данных программа формирует SQL-запрос для записи нового студента в БД с соответствующими параметрами и шаблоном отпечатка пальца. Также эти данные загружаются в таблицу приложения. При редактировании студента загружается то же диалоговое окно, только с уже загруженными данными по выбранному студенту. При удалении программа формирует SQL-запрос по удалению записи студента по заданным параметрам, а также выбранный студент удаляется из таблицы приложения.

Пользователь также может добавлять и удалять предметы из базы данных. При добавлении нового предмета пользователю предоставляется выбор, какому преподавателю привязать данный предмет, а при удалении программа удаляет все записи, включающие в себя данный предмет, такие как посещаемость и учебный план.

ЛИТЕРАТУРА

1. Примеры SQL-запросов к базе данных MySQL [Электронный ресурс]. – 2017. – Режим доступа: <http://tradebenefit.ru/primery-mysql-zaprosov>. – Дата доступа: 09.05.2017.
2. Работа с COM-портом в Java [Электронный ресурс]. – 2017. – Режим доступа: <https://ru.stackoverflow.com/questions/401234/java-%D0%A7%D1%82%D0%B5%D0%BD%D0%B8%D0%B5-%D1%81-com-%D0%BF%D0%BE%D1%80%D1%82%D0%B0>. – Дата доступа: 10.05.2017.

УДК 004.457

ОСОБЕННОСТИ АППАРАТНО-ПРОГРАММНОГО ВЗАИМОДЕЙСТВИЯ СИСТЕМЫ КОНТРОЛЯ ПОСЕЩАЕМОСТИ СТУДЕНТОВ НА ОСНОВЕ АНАЛИЗА ОТПЕЧАТКОВ ПАЛЬЦЕВ

К.А. АПАНЧЁНОК
(Представлено: М.В. МАТЮШ)

Рассматривается работа устройства системы контроля посещаемости студентов с использованием отпечатков пальцев. Представлен способ взаимодействия клиентского приложения для компьютера под управлением Windows и устройства на платформе ArduinoUno. Дано подробное описание передачи данных от приложения к устройству и обратно.

В разрабатываемом приложении и проектированном устройстве требовалось осуществить передачу данных между собой. В качестве канала передачи данных использовался последовательный СОМпорт. В Arduino имеется уже встроенный преобразователь USB-COM, поэтому работа устройство принимает и отправляет данные посредством СОМпорта [1].

Устройство по умолчанию находится в режиме ожидания команды. Всего устройство может обрабатывать набор необходимых для полноценной работы команд (табл. 1). Каждая команда предполагает дальнейшее взаимодействие со сканирующим устройством, т.е. обмен данными между Arduino и сканером отпечатков. Для того чтобы сократить время передачи и обработки команды, было решено использовать номера в качестве команд, вместо их названия.

Таблица 1. – Команды и их функции

Команда	Функция
1	Получение кода отпечатка пальца
2	Передача и запись кода отпечатка пальца в сканер
3	Сканирование пальца и поиск совпадения
4	Очистка БД сканера

При выполнении *первой команды* на сканер отправляется команда считывания отпечатка пальца и записи результата в первый буфер сканера, затем снова команда считывания отпечатка пальца и запись во второй буфер. После отправляется команда на создание усредненного шаблона отпечатка (для того, чтобы ввести погрешность при последующем поиске совпадений с данным отпечатком) и в конце – команда передачи шаблона со сканера на Arduino и команда передачи шаблона с Arduino в приложение.

При выполнении *второй команды* Arduino ожидает приема кода отпечатка пальца определенной длины и номера для записи шаблона в память сканера. Затем Arduino отправляет команду передачи шаблона на сканер и номера для записи в необходимую ячейку памяти.

При выполнении *третьей команды* на сканер отправляется команда для сканирования отпечатка, записи его в буфер и поиск совпадения в базе сканера. Сканер, в свою очередь, отправляет обратно результат, который потом дублируется на отправку в приложение.

При выполнении *четвертой команды* на сканер отправляется команда, которой устройство очищает энергонезависимую память сканера, а также временные буферы.

Таким образом, пользовательское приложение отправляет команду и ожидает приема метасимволов, означающих выполнение команды, а затем продолжает работу с пользователем (псевдокод 1).

Псевдокод 1 – функция отправки команды поиска совпадения

```
String findMatch() {
    String ret = null;
    try {
        data2 = "";
        serialPort.writeString("3\n");    //отправкакоманды 3
        while (true) {sleep(1);
            data = serialPort.readString();    //чтениеответныхданных
            if (scan == 0) { break;}
            if (data != null) {data2 += data;}
```



```

        if (data2.contains("$#")) {           //проверка на наличие мета-символов
            int a = data2.indexOf("ID=") + 3;
            int b = data2.indexOf("$");
            ret = data2.substring(a, b); //получениеномерасовпадения
            break; }
        if (data2.contains("Did not find a match")) { break;} //проверкананесовпадение
    }
} catch (SerialPortException ex) { Logger.getLogger(Window.class.getName()).log(Level.SEVERE,
null, ex);
} catch (InterruptedException ex) {Logger.getLogger(Window.class.getName()).log(Level.SEVERE,
null, ex);}
data2 = ""; //очистка буфера приема данных
return ret;} //вывод результата функции

```

На устройстве Arduino обрабатываемая команда выполняется и выдает результат как при нахождении совпадения (рис. 1), так и при его отсутствии (рис. 2).

```

FingerprintTest...#
Success!#
SWITCH MODE#
SWITCH = 3
MatchMode#
Scan finger#
Taked#
Converted#
Found a print match!#
Found ID=1$#
SWITCH MODE#

```

Рисунок 1. – Результат выполнения команды поиска совпадений при нахождении отпечатка в БД

```

SWITCH MODE#
SWITCH = 3
MatchMode#
Scan finger#
Taked#
Converted#
Did not find a match#
SWITCH MODE#

```

Рисунок 2. – Результат выполнения команды поиска отпечатка при отсутствии совпадения

Разработанное устройство выполнено на базе аппаратной платформы ArduinoUno и сканера отпечатков пальцев ZFM-20 [2]. Данный сканер позволяет получать шаблоны отпечатков за время < 1 секунды, хранить ввстроенной памяти до 1600 значений. Аппаратная часть представляет собой комплекс из двух этих устройств, которые взаимодействуют между собой и связываются с программой на ПК. Связь между ПК и устройством осуществлена посредством COM-порта [3]. Сканер по умолчанию настроен на прием и передачу информации в шестнадцатичном коде путем формирования пакетов данных (табл. 2).

Таблица 2. – Формат построения пакета данных

Заголовок (2 байта) Обычно значение EF01H	Адрес (4 байта)	Идентификатор пакета данных (1 байт) 01H – командный; 02H – пакет данных; 07H – ответный пакет; 08H – завершающий пакет данных	Размер пакета (2 байта)	Содержание пакета (размер задается в настройках сканера. Максимум 256 байт)	Контрольная сумма (2 байта)

Для корректной и более удобной работы со сканером использована библиотека AdufruitFingerPrintLib [4], которая позволяет использовать большинство функций данного сканера без формирования пакетов данных вручную. На Arduino реализованы функции, необходимые для обмена данными с ПК и сканером.

При включении Arduino происходит авторизация сканера (инициализация порта обмена данными со сканером и проверка его на готовность работы) и подключения с ПК (открытие порта для обмена данными с ПК), затем идет ожидание получения одной из команд от ПК, перечисленных выше. Рассмотрим каждую функцию подробнее в рамках устройства.

Функция получения шаблона отпечатка пальца при выполнении отправляет на сканер команду начала сканирования и ожидает сканирования отпечатка. После того как палец был просканирован, его скан загружается в один из двух буферов сканера для дальнейшей обработки. Затем сканер снова переходит в режим сканирования и записывает второй скан в другой буфер. Это необходимо для построения усредненного значения шаблона отпечатка пальца. Такой шаблон будет распознаваться сканером даже при неполном соответствии (70–80%), не допуская при этом ошибок при поиске отпечатка в БД. После составления шаблона на сканер подается команда для выгрузки шаблона со сканера на Arduino, которая, в свою очередь, отправляет его через COM-порт на ПК (псевдокод 2).

Псевдокод 2 – функция получения шаблона отпечатка пальца со сканера

```
int cas1(){
  Serial.println("GetCharFromIMG mode");
  Serial.println("Scan finger#");
  int p = -1;
  p = GetImage(1); //отправка команды получения скана пальца
  if(p != -1){
    delay(2);
    p = ImageToTz(1); //конвертирование изображения в массив символов и запись в буфер №1
    delay(1000);
    while (p != FINGERPRINT_NOFINGER) {
      p = finger.getImage(); //отправка команды получения скана пальца для проверки на наличие
//пальца на сканере (проверка ввода)
    }
    p = GetImage(1); //отправка команды получения скана пальца
    if(p != -1){
      delay(2);
      p = ImageToTz(2); //конвертирование изображения в массив символов и запись в буфер № 2
      p = createModel(); //создание усредненного шаблона отпечатка пальцев при совпадении
//более 90% (то есть при скане одного и того же пальца) и запись результата в буфер № 1
      if(p == 0)
      {
        downloadModel(1); //если создание успешно, то отправляется команда получения значения
//из буфера № 1
      }
      else{
        Serial.println("NOTMATCH"); } } //иначе, вывод сообщения «НЕТ СОВПАДЕНИЯ»
```

Функция добавления шаблона отпечатка во внутреннюю память устройства при вызове запускает прием данных шаблона отпечатка пальца с ПК, затем принимает уникальный идентификатор данного отпечатка. Затем запускается функция отправки шаблона отпечатка в буфер сканера. После выполняется функция записи данного отпечатка в память по указанному ID (псевдокод 3).

Псевдокод 3 – функция добавления шаблона отпечатка в память

```
int cas2(void){
  Serial.println("DownloadCharToCharbuf mode");
  uint8_t packet[688];
  Serial.println("Start recive char...#");
  intp = 0;
  p = reciveChar(packet); //получение данных с COM-порта от ПК
  Serial.println("End recive#");
```

```
Serial.println("Enter ID#");
int id = readnumber();    //получениеидентификатора
Serial.print("ID=");
Serial.println(id);
Serial.println("Upload#");
uploadModel(1, packet);   //передачашаблонаотпечаткапальцавбуфер №1
Serial.println("Storing#");
storeModel(id);           //записьзначениябуфера №1 впамятьподномеромid
}
```

Функция поиска совпадений по скану при выполнении отправляет команду захвата отпечатка пальца со сканера. Затем при наличии значения пальца конвертируется в буфер № 1, и отправляется команда получения номера совпадения (псевдокод 4).

Псевдокод 4 – поиск совпадения в памяти устройства

```
int cas3(void){
Serial.println("MatchMode#");
Serial.println("Scan finger#");
int p = -1;
p = GetImage2(); //получение скана пальца
if(p != -1){
delay(2);
p = ImageToTz(1);    //конвертирование изображения в символьное представление
getMatch();           //поиск совпадения о сходстве отпечатка с уже имеющимися в памяти
}
}
```

Функция очистки памяти сканера предназначена для очистки шаблонов перед загрузкой новых шаблонов какой-либо группы (псевдокод 5).

Псевдокод 5 – функция очистки памяти сканера

```
intcas4(void){
Serial.println("EmptyDataBase");
emptyBase();
}
```

ЛИТЕРАТУРА

1. Работа с COM-портом Arduino [Электронный ресурс]. – 2017. – Режим доступа: <http://arduino.ru/forum/programirovanie/rabota-s-com-portom>. – Дата доступа: 10.05.2017.
2. ZFM-20dingerprintmoduledatasheet [Электронный ресурс]. – 2017. – Режим доступа: https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library/blob/master/documentation/ZFM-20_Fingerprint_Module.pdf. – Дата доступа: 10.05.2017.
3. Работа с COM-портом в Java [Электронный ресурс]. – 2017. – Режим доступа: <https://ru.stackoverflow.com/questions/401234/java-%D0%A7%D1%82%D0%B5%D0%BD%D0%B8%D0%B5-%D1%81-com-%D0%BF%D0%BE%D1%80%D1%82%D0%B0>. – Дата доступа: 10.05.2017.
4. Arduino library for interfacing to the fingerprint sensor [Электронный ресурс]. – 2017. – Режим доступа: <https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library/>. – Дата доступа: 10.05.2017.

УДК 681.5:316.7(476)

**ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ
СИСТЕМЫ УПРАВЛЕНИЯ АФИШЕЙ КУЛЬТУРНЫХ СОБЫТИЙ РЕСПУБЛИКИ БЕЛАРУСЬ****В.Ю. ДЕМЕНТЬЕВ****(Представлено: Ю.Н. КРАВЧЕНКО)**

Демонстрируется разработка системы управления афишей культурных событий Республики Беларусь. Показано создание системы для удобного управления афишей культурных событий Республики Беларусь. Рассматриваются вопросы, связанные с процессом проектирования базы данных системы управления.

Любое приложение, будь то web-приложение, desktop или мобильное, которое нацелено на предоставление актуальной информации, нуждается в администрировании, т.е. управлении контентом в удобной для администратора форме. Обычно для таких потребностей используются административные панели, которые позволяют обращаться к базе данных посредством web-форм, изменять содержимое приложений в визуальных редакторах и т.д. Поэтому основное преимущество разрабатываемой системы управления в том, что администратору не нужно редактировать базу данных напрямую, а из-за необходимости хранения множества различных сущностей и большого объема данных проектируемая база данных должна быть наиболее простой, нормализованной и эффективной.

Для определения сущностей, хранимых в базе данных, необходимо выделить функции, которые будут доступны в системе управления. *Во-первых*, стоит отметить, что система будет разделена на две части: 1) *глобальная система управления* (для управления и моделирования мест проведения событий (театры и кинотеатры, дома культуры, открытые концертные площадки и т.д.), для управления администраторами и т.п.); 2) *система управления афишей места проведения культурного мероприятия* (для непосредственного управления афишей). *Во-вторых*, из-за разделения системы будет выделено две роли – глобальный администратор и администратор проведения мероприятия.

Таким образом, *администратору по месту проведения* будет доступен небольшой ряд функций по работе с афишей событий в непосредственно выделенном для них месте и редактированию информации о самом месте:

- добавление/редактирование/удаление мероприятий (событий);
- добавление/редактирование/клонирование/удаление цен;
- добавление/удаление сеансов проведения мероприятий;
- редактирование информации о месте проведения событий.

Глобальный администратор будет иметь намного больше прав:

- добавление/редактирование/удаление мероприятий для любого места проведения события;
- добавление/редактирование/клонирование/удаление цен для любого места проведения события;
- добавление/удаление сеансов для любого места проведения события;
- редактирование информации о любом месте проведения мероприятия;
- редактирование информации любого администратора;
- добавление новых мест для проведения событий в систему;
- регистрация новых администраторов для мест проведения мероприятий;
- просмотр лога изменений любого администратора;
- черный список администраторов;
- просмотр различной статистики по местам проведения событий.

С учетом выдвинутых к системе управления функций, можно выделить следующие *сущности проектируемой базы данных*:

- администратор (user) – пользователь системы управления, обладающий определенными правами редактирования данных места проведения мероприятий;
- место проведения события (place) – заведение, в котором проводятся сеансы мероприятий;
- город (city) – город, в котором происходит событие;
- мероприятие (action) – событие, происходящее в назначенное время и в назначенном месте;
- тип события (type) – тип мероприятия (кинофильм, концерт и любые другие массовые события);
- сеанс (session) – единица проведения события. Информация о сеансе включает в себя стоимость сеанса, дату и время действия мероприятия;
- цена (cost) – стоимость одного сеанса;
- архив (archive) – список событий, которые завершились;
- лог (log) – запись об изменении в системе, которые внесены администратором места проведения событий.

Чтобы построить схему реляционной базы данных, необходимо определить совокупность отношений, которые составляют базу данных. Необходимо, чтобы выполнялось условие целостности отношений: в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа, т.е. любая запись любого отношения должна быть отличной от любой другой записи этого отношения.

В проектируемой базе данных все таблицы будут иметь первичные ключи и связаны между собой связью один ко многим.

Схема спроектированной базы данных, нормализованной до 3-ей нормальной формы, представлена на рисунке 1.

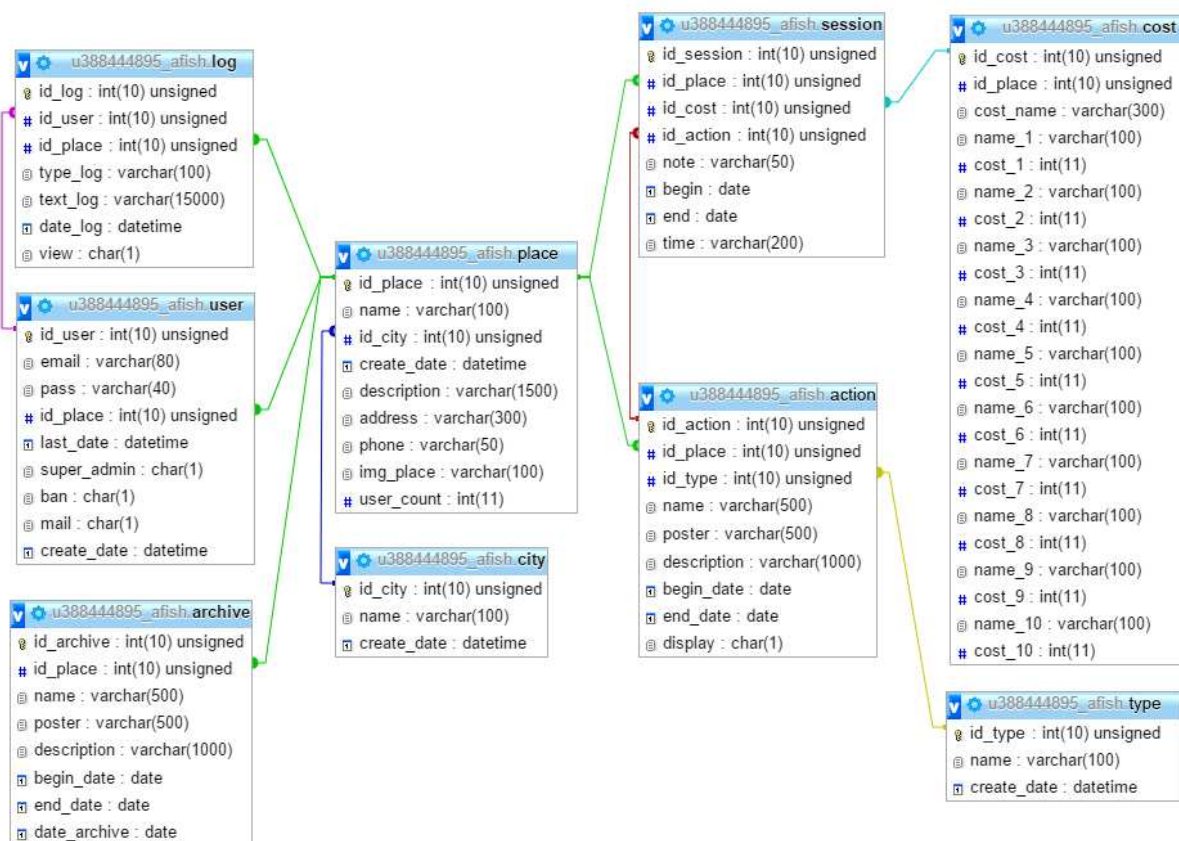


Рисунок 1. – Схема спроектированной базы данных

В качестве СУБД была выбрана MySQL. СУБД MySQL обладает высокой скоростью работы по сравнению с другими системами управления. Также преимуществом MySQL является то, что это многопользовательская система, она не налагает ограничений на количество пользователей, одновременно работающих с базой данных.

MySQL может хранить огромное количество записей в своих таблицах. Этим обеспечивается возможность создания крупных проектов. Немаловажным показателем является также простая и эффективная система безопасности MySQL.

Для удобного формирования архива событий необходимо создать триггер (after delete) на удаление мероприятия.

Заключение. В результате проделанной работы спроектирована база данных, при помощи которой можно создать систему управления афишей культурных событий для удобной работы администратора. Также использование базы данных делает удобным хранение и управление данными афиши.

ЛИТЕРАТУРА

1. Кузнецов, М.В. MySQL 5 / М.В. Кузнецов, И.В. Симдянов. – СПб. : БХВ-Петербург, 2010. – 1024 с.
2. Малыгина, М. Базы данных: основы, проектирование, использование / М. Малыгина. – М. : БНУ, 2004. – 512 с.
3. Фестиваль педагогических идей «Открытый урок» [Электронный ресурс] / ИД «Первое сентября». – М., 2014. – Режим доступа: <http://festival.1september.ru>. – Дата доступа: 29.03.2014.

УДК 316.7:316.7(476)

**ПРОЕКТИРОВАНИЕ КЛИЕНТСКОЙ ЧАСТИ СИСТЕМЫ УПРАВЛЕНИЯ
АФИШЕЙ КУЛЬТУРНЫХ СОБЫТИЙ РЕСПУБЛИКИ БЕЛАРУСЬ****В.Ю. ДЕМЕНТЬЕВ,**
(Представлено: Ю.Н. КРАВЧЕНКО)

Рассматриваются вопросы, связанные с процессом проектирования клиентской части системы управления афишей культурных событий Республики Беларусь. Цель – создание системы для удобного управления афишей культурных событий. Использование спроектированной системы позволит облегчить процесс управления афишей любых культурных событий и мероприятий.

Цель разработки клиентской части системы управления, в первую очередь, состоит в удобстве работы администратора с записями из базы данных и исключение возможных ошибок при работе с базой данных. Таким образом, особое внимание необходимо уделить разработке графического пользовательского интерфейса.

Если рассматривать системы управления в целом, то можно выделить несколько минусов. Как правило, все CMS разрабатываются под какие-то общие потребности (добавление статей, модерирование пользователей и др.) и возникают определенные трудности при необходимости добавления каких-либо специфических и уникальных для таких CMS модулей. Из-за этого вся система становится более громоздкой. Некоторые возможности системы управления не используются, так как они не нужны для текущих потребностей пользователя, но на них тратятся определенные ресурсы. Кроме этого, стиль таких CMS недостаточно современен. На данный момент Google задает тенденцию в так называемом Material Design, схожим со стилем минимализм. В проектируемой клиентской части системы управления будет использован данный стиль, так как он все больше набирает популярность и выглядит достаточно строго и аккуратно. Нет необходимости в красочных изображениях, из-за которых пользователь может теряться в системе и не воспринимать или упускать важную информацию. Вся панель администратора должна быть выполнена в едином приятном для глаза стиле. Также стоит отметить, что многие системы управления не всегда удобны. Для выполнения основных задач бизнес-процесса необходимо выполнить неочевидные шаги для пользователя. Разрабатываемая система должна быть совершенно понятна и прозрачна, чтобы любой администратор места проведения культурного мероприятия мог интуитивно выполнить любое действие путем нажатия не более двух кнопок. Отдельное внимание будет уделено эргономике. Удобное размещение элементов также позволит облегчить работу администратора и снизить когнитивную нагрузку.

В качестве среды разработки был выбран JetBrais PhpStorm 8, так как данная среда разработки имеет отличный интерфейс и поддержку самых последних версий PHP. Кроме этого, PhpStorm предоставляет отличные возможности для удобной разработки на языках JavaScript, HTML и CSS.

За основу разработки дизайна была взята стилистика CSS библиотеки Skeleton. Данная библиотека предоставляет минимальные возможности для начала разработки собственного проекта, поэтому все основные стили будут написаны собственноручно.

Для отображения иконок в приложении будет использоваться библиотека Font Awesome. Это векторные иконки, которые интегрированы в шрифт. Их преимущество в том, что они способны изменять размер без потери качества, а также к ним можно применять различные CSS стили, например, изменение цвета.

Для удобной работы с JavaScript будет использоваться библиотека JQuery. Она предоставляет простой доступ к DOM-элементам страницы. Большинство POST/GET запросов при работе с данными в панели управления будут обрабатываться посредством технологии AJAX. Таким образом, это уменьшит время ожидания получения ответов от сервера, так как подгружаться на странице будут только необходимые данные, остальные данные затрагиваться не будут.

Для построения графической составляющей необходимо определить, какие функции должна выполнять система управления. Прежде всего, отметим, что система будет разделена на две части – *глобальную систему управления* (для управления и моделирования мест проведения событий (театры и кинотеатры, дома культуры, открытые концертные площадки и т.д.) и *систему управления афишей места проведения культурного мероприятия* (для непосредственного управления афишей).

Из-за разделения системы выделено две роли – *глобальный администратор* и *администратор проведения мероприятия*.

Администратору по месту проведения будет доступен небольшой ряд функций по работе с афишей событий в непосредственно выделенном для них месте и редактированию информации о самом месте:

- добавление/редактирование/удаление мероприятий (событий);

- добавление/редактирование/клонирование/удаление цен;
- добавление/удаление сеансов проведения мероприятий;
- редактирование информации о месте проведения событий;

Глобальный администратор будет иметь намного больше прав:

- добавление/редактирование/удаление мероприятий для любого места проведения события;
- добавление/редактирование/клонирование/удаление цен для любого места проведения события;
- добавление/удаление сеансов для любого места проведения события;
- редактирование информации о любом месте проведения мероприятия;
- редактирование информации любого администратора;
- добавление новых мест для проведения событий в систему;
- регистрация новых администраторов для мест проведения мероприятий;
- просмотр лога изменений любого администратора;
- черный список администраторов;
- просмотр различной статистики по местам проведения событий.

Все страницы будут сверстаны по единому шаблону: шапка – основной контент – подвал.

Прежде чем начать работу с системой управления, необходимо пройти процедуру авторизации. Это выполняется посредством формы входа, которая состоит из двух полей для ввода e-mail адреса и пароля, а также кнопки «Войти».

Далее будут описаны страницы, доступные глобальному администратору.

В шапке сайта располагается меню, в котором находятся следующие пункты меню:

- статистика;
- управление;
- лог изменений;
- выход.

В свою очередь, пункт меню «Управление» может быть раскрыт в виде выпадающего списка, который включает в себя следующие пункты:

- управление местами проведения событий;
- управление пользователями;
- управление форматами мероприятий;
- управление списком городов.

На странице статистики будет расположена различная информация по количеству сущностей в системе, статистика по местам проведения мероприятий и т.п.

На странице «Лог изменений» будет расположен список изменений, которые внесли в систему администраторы мест проведения событий. Глобальный администратор сможет следить за всеми изменениями и принимать меры в случае нарушения каких-либо правил.

Данные страницы состоят из списка сущностей, некоторой информации о них и действий, которые для них возможны. Используя данную страницу, администратор может добавить новое место проведения мероприятия в систему, отредактировать или удалить существующее, либо войти в любое из мест событий от лица другого администратора и выполнять любые действия, доступные обычному администратору проведения событий. На странице управления пользователями администратор также может добавлять новых администраторов, удалять или редактировать существующих, а также добавлять их в черный список.

Страницы управления форматами фильмов и списком городов также похожи. На них отображается список сущностей с возможностью редактирования и удаления, а также дополнительное поле быстрого добавления новой сущности. Все действия выполняются посредством технологии AJAX.

Так как у администратора места проведения события другие функции, в его панели управления располагаются иные пункты меню:

- афиша;
- добавить;
- архив;
- в панель управления;
- настройки;
- выход.

При этом пункт меню «Добавить» включает в себя следующие подпункты, которые раскрываются в виде выпадающего списка:

- добавить событие;
- добавить сеансы;
- добавить цены.

На странице «Афиша» располагается список текущих событий, которые проводятся в текущем месте, а также его сеансы и возможные действия. На данной странице администратор может сделать ме-

роприятие невидимым для конечных пользователей, перейти к редактированию события или удалить его. После удаления событие помещается в архив и весь список удаленных мероприятий доступен на соответствующей странице. В архиве администратор может удалить событие полностью из системы.

Пункт меню «В панель управления» отображается только, если в панель управления вошел глобальный администратор через страницу «Управление местами проведения событий». Она возвращает администратора обратно к списку мест проведения событий, зарегистрированных в системе. Для обычного администратора данный пункт недоступен.

На странице настроек администратор может изменять данные о месте проведения событий – телефон, адрес или описание. Также там можно сменить пароль.

Страницы добавления и редактирования событий похожи и состоят из одинаковых полей. На них есть возможность загрузки рекламного изображения для планируемого события и выбор формата мероприятия. Кроме этого на данных страницах указывается подробное описание проводимого события.

На странице добавления цен располагается список уже добавленных цен в систему и форма для добавления новой цены. При этом цены могут быть составные, т.е. одна запись может содержать до 10 цен. Таким образом, если на месте проведения мероприятия существует разделение зала на сектора, с разной стоимостью, администратор в пределах одной записи может указать стоимость каждого сектора. На той же странице администратор может удалить, отредактировать или создать клон записи.

На странице добавления сеансов располагаются выпадающие списки с доступными событиями и ценами, которые необходимо выбрать для добавления сеанса. Там же необходимо указать период и время сеанса.

Кроме описанных пунктов меню, в системе в подвале каждой страницы доступны следующие пункты меню:

- помощь;
- правила системы;
- написать администратору.

На странице помощи располагается список возможных проблем и методы их решения в виде видеозаписи, а также примеры использования всех доступных администратору функций. На странице правил системы располагается список правил, из-за невыполнения которых администратор может попасть в черный список системы. А при нажатии ссылки «Написать администратору» пользователю отобразится всплывающее окно с формой для отправки сообщения администратору.

Заключение. Интерфейс является понятным пользователю без обучения работы с системой управления афишей. Присутствует очень подробная страница помощи в случае, если у пользователя возникли трудности. В случае выполнения неверных действий пользователю будет показано сообщение об ошибке, а в случае корректного выполнения – сообщение об успешном завершении операции.

Использование спроектированной системы позволит облегчить процесс управления афишей любых культурных событий и мероприятий.

ЛИТЕРАТУРА

1. Спецификация Google – Material design [Электронный ресурс] / Google. Material design. – Режим доступа: <http://www.google.com/design/spec/>. – Дата доступа: 29.09.2015.
2. Макфарланд, Д. Большая книга CSS3 / Д. Макфарланд. – 3-е изд. – СПб. : Питер, 2014. – 608 с.

УДК 004.624

ИСПОЛЬЗОВАНИЕ LIQUIBASE
ДЛЯ УПРАВЛЕНИЯ ИЗМЕНЕНИЯМИ СХЕМЫ БАЗЫ ДАННЫХ

Д.Ю. ДЁКИН

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматривается платформно-независимая библиотека с открытым исходным Liquibase, при помощи которой можно осуществлять контроль за изменениями схемы базы данных. Проанализированы некоторые возможности библиотеки Liquibase. Представлены базовые примеры, демонстрирующие простоту применения данной технологии в рамках управления изменениями структуры базы данных.

При разработке большинства приложений используются различные системы контроля версий (Git, Subversion и т.п.). Возникает вопрос, как с течением времени меняется схема БД, как это отслеживается, чем управляется? Случается, что изменения базы данных на тестовых и «боевых» серверах делаются вручную. Данный подход может не вызывать сложностей, если над проектом работает один или два разработчика, но, если работает целая команда, становится трудно синхронизировать все изменения между разработчиками.

Liquibase – платформно-независимая библиотека с открытым исходным кодом (реализована на Java), которая предназначена для управления изменениями схемы БД, а также является хорошим инструментом в миграции с одной базы данных на другую.

Чтобы начать работать с *Liquibase*, требуется указать несколько настроек для библиотеки:

1. Драйвер базы данных.
2. Путь к драйверу.
3. Url к базе данных.
4. Логин и пароль к базе данных.
5. Путь к чейнджлогу.

Основными понятиями, которыми оперирует *Liquibase*, являются:

1. Чейнджсет (от англ. changeset – набор изменений).
2. Чейнджлог (от англ. changelog – файл, состоящий из набора чейнджсетов).

Таким образом, отдельный чейнджлог представляет собой совокупность чейнджсетов в рамках какой-то отдельно взятой таблицы базы данных.

Код чейнджсетов может быть описан при помощи XML, JSON, YAML, SQL или даже на groovy. Например, при помощи следующего кода (листинг 1) можно создать таблицу **person** с двумя полями: *id*, *name*.

Листинг 1 – Создание таблицы «person»

```
<databasechangelog
  xmlns=http://www.liquibase.org/xml/ns/dbchangelog xmlns:xsi=http://www.w3.org/2001/XMLSchema-
instance xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-2.0.xsd">
  <changeset id="2017-09-09-00" author="dzmitry@mail.com">
    <createtable tablename="person">
      <column name="id" type="BIGINT" autoincrement="true">
        <constraints nullable="false" primarykey="true"/>
      </column>
      <column name="name" type="VARCHAR(255)">
        <constraints nullable="false"/>
      </column>
    </createtable>
  </changeset>
</databasechangelog>
```

Liquibase имеет встроенную поддержку rollback (отмены) для некоторых типов чейнджсетов, как, например, для чейнджсета выше (*createTable*). Другие чейнджсеты автоматически быть откатенными не могут. В таких случаях предусмотрена секция *rollback*, позволяющая описывать действия для отмены изменений чейнджсета (листинг 2).

Листинг 2 – Пример чейнджсета с секцией «rollback»

```
<databasechangelog
  xmlns=http://www.liquibase.org/xml/ns/dbchangelog xmlns:xsi=http://www.w3.org/2001/XMLSchema-
instance xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-2.0.xsd">
  <changeset id="2017-09-09-00" author="dzmitry@mail.com">
    <insert tablename="person">
      <column name="id" value="John Smith"/>
    </insert>
    <rollback>
      DELETE FROM person WHERE name LIKE 'John Smith';
    </rollback>
  </changeset>
</databasechangelog>
```

Чейнджсеты в *Liquibase* определяются уникальностью комбинации тегов `id`, `author`, а также имени файла чейнджлога. Таким образом, чтобы успешно применять чейнджсеты (в рамках одного чейнджлога), необходимо задавать уникальное поле для `id` и `author`.

На практике же (чаще всего) в качестве `id` берется значение текущей даты, и в конец добавляется порядковый номер чейнджсета. Поэтому если бы в чейнджлог из листинга 2 нужно было бы добавить еще один чейнджсет, то его `id` имело бы значение «2017-09-09-01».

Имея несколько чейнджлогов, можно создать один кумулятивный файл, который будет импортировать другие чейнджлоги (листинг 3).

Листинг 3 – Пример кумулятивного чейнджлога

```
<databasechangelog
  xmlns=http://www.liquibase.org/xml/ns/dbchangelog>
  <include file="v-1.0/person-v-0-1.xml" relativeToChangelogFile="true"/>
  <include file="v-1.0/data-v-0-1.xml" relativeToChangelogFile="true"/>
</databasechangelog>
```

Используя *Liquibase*, стоит следовать ряду правил и рекомендаций, который облегчит управление версиями схемы базы данных в дальнейшем. Итак, версионность приложения должна быть отражена в структуре папок чейнджлогов:

1. Чейнджлоги изменения базы данных должны содержаться в отдельных папках, соответствующих версиям приложения
2. Держать кумулятивный файл чейнджлогов для каждой версии.
3. Соблюдать правила именования чейнджлогов.
4. Лучше много маленьких чейнджсетов, чем один большой.
5. Чейнджлоги, изменяющие схему базы данных, должны быть атомарными.

Liquibase оперирует значениями хеша чейнджсета. Поэтому если чейнджсет уже был применен, но оказалось, что нужно добавить что-то еще, то должен создаваться отдельный чейнджсет. Иначе, если отредактировать имеющийся чейнджсет, изменится хеш данного чейнджсета. В таком случае *Liquibase* не применит данные изменения к базе данных.

Заключение

Рассмотрены некоторые возможности библиотеки *Liquibase*, а также даны советы и рекомендации по организации и хранению чейнджсетов. Предоставлены базовые примеры, демонстрирующие простоту применения данной технологии в рамках управления изменениями структуры базы данных.

ЛИТЕРАТУРА

1. Хабрахбр [Электронный ресурс] Использование Liquibase без головной боли. 10 советов из опыта реальной разработки. – Режим доступа: <https://habrahabr.ru/post/179425>. – Дата доступа: 24.09.2017.
2. Хабрахбр [Электронный ресурс] Управление миграциями БД с Liquibase. – Режим доступа: <https://habrahabr.ru/post/179425>. – Дата доступа: 24.09.2017.
3. Ezcode [Электронный ресурс] Версионирование структуры БД с помощью Liquibase. – Режим доступа: <http://easy-code.ru/lesson/database-versioning-liquibase>. – Дата доступа: 24.09.2017.

УДК 004.657

МУВАТИС КАК ИНСТРУМЕНТ ДЛЯ РАБОТЫ С БАЗАМИ ДАННЫХ
В JAVA-ПРИЛОЖЕНИЯХ

Д.Ю. ДЁКИН

(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматривается MyBatis фреймворк для отображения данных из базы на доменные объекты в Java-приложениях. Представлен краткий обзор базовых возможностей MyBatis фреймворка и его базовой конфигурации. Приведены примеры маппингов и запросов к базе данных, вызова запросов из Java-кода.

При проектировании и разработке приложений практически всегда приходится иметь дело с базами данных. Поэтому встает вопрос о выборе того или иного ORM или persistence фреймворка, который будет служить для отображения данных из базы в пользовательские объекты. Существует достаточно большое количество различных ORM, таких как Hibernate, JPA, EclipseLink и др. Каждый из них имеет свои достоинства и недостатки. Большинство фреймворков поддерживают JPA спецификацию.

Рассматриваемый в данной статье фреймворк является альтернативой JPA спецификации.

MyBatis является persistence фреймворком, который отображает SQL запросы на доменные объекты. Данный фреймворк (так же, как и ORM фреймворки) ликвидирует необходимость писать шаблонный и ненужный код для уже давно решенных задач. MyBatis позволяет отображать данные из базы в примитивные Java-типы и POJO (Plain Old Java Object, простой Java-объект). MyBatis может быть сконфигурирован как при помощи XML, так и при помощи аннотаций. Для удобства и простоты последующие примеры будут приводиться на XML.

MyBatis хорошо подходит для тех случаев, когда в приложении имеется достаточно много различных сложных выборок из нескольких таблиц (много JOIN-ов). Все дело в том, что MyBatis позволяет отображать SQL-запросы на сущности. Поэтому производительность запроса полностью определяется разработчиком. Это позволяет полностью контролировать, какие запросы и как именно будут отправляться в БД.

Для базовой конфигурации необходимо указать источник данных (от англ. data source) с именем драйвера и url к БД, логином и паролем. Также в конфигурационном файле указываются type alias (доменные объекты), mapper (xml-файлы, которые хранят SQL-запросы). Пример приведен в листинге 1.

Листинг 1 – Пример конфигурационного файла

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <properties resource="config.properties">
    <property name="schema" value="db_schema"/>
  </properties>

  <dataSource type="POOLED">
    <property name="driver" value="${driver}"/>
    <property name="url" value="${url}"/>
    <property name="username" value="${username}"/>
    <property name="password" value="${password}"/>
  </dataSource>

  <typeAliases>
    <typeAlias type="by.psu.domain.User"/>
    <typeAlias type="by.psu.domain.Role"/>
  </typeAliases>

  <mappers>
    <mapper resource="by/psu/repository/IUserMapper.xml"/>
    <mapper resource="by/psu/repository/IRoleMapper.xml"/>
  </mappers>
</configuration>
```

Для того чтобы показать, как происходит отображение результатов выборки на доменные объекты, введем следующие простые типы: User, Role (методы доступа к полям опущены для экономии места). Данные классы представлены в листинге 2.

Листинг 2 – Описание классов User и Role

```
class User {  
    private String id;  
    private String name;  
    private List<Role> roles;  
}  
  
class Role {  
    private String id;  
    private String name;  
}
```

Теперь для данной простой иерархии классов введем так называемые маппинги (отображения), представленные в листинге 3.

Листинг 3 – Пример маппинга на доменные классы

```
<resultMap type="User" id="userMap">  
    <result id="id" column="u_id"/>  
    <result property="name" column="u_name"/>  
    <collection property="roles" ofType="Role" resultMap="roleMap"/>  
</resultMap>  
  
<resultMap type="Role" id="roleMap">  
    <result id="id" column="r_id"/>  
    <result property="name" column="r_name"/>  
</resultMap>
```

Как можно видеть из листинга, маппинги довольно просты.

Таким образом, мы задаем правило того, как должны будут заполняться наши объекты из выборки данных. MyBatis будет брать значение колонки и вставлять его в объект при помощи соответствующего setter-а (метода установки значения).

В данном примере используется коллекция ролей. Однако если есть необходимость реализовать ассоциацию (один-к-одному связь), то в таком случае в маппере будет использоваться **association** тег.

Идентификаторы мапперов должны быть уникальными (аттрибут *id* в теге *resultMap*).

Для наглядности на рисунке 1 представлена схема БД для данных сущностей.

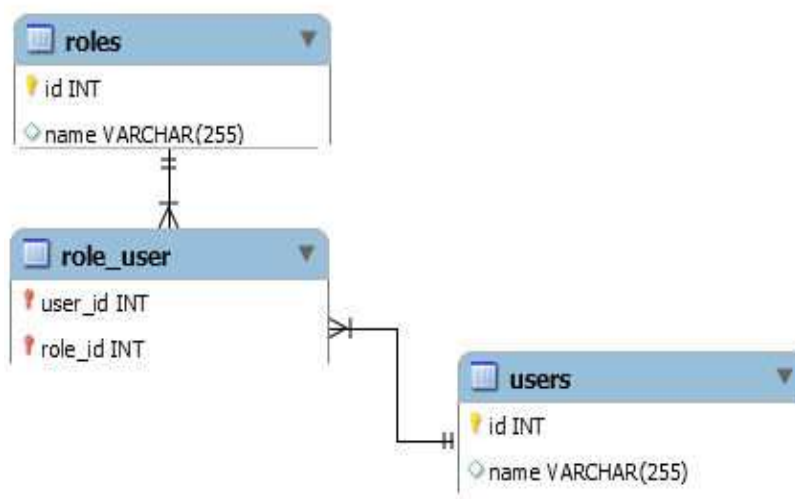


Рисунок 1. – Схема базы данных

Можно привести пример обычного запроса, который должен вернуть пользователя со всеми его ролями по его id (листинг 4).

Листинг 4 – Пример запроса выборки данных

```
<select id="getUserById" resultMap="userMap">
  SELECT
    u.id AS u_id,
    u.name AS u_name,
    r.id AS r_id,
    r.name AS r_name
  FROM users AS u
  INNER JOIN user_role AS ur
    ON u.id = ur.user_id
  INNER JOIN roles AS r
    ON ur.role_id = r.id
  WHERE u.id = #{id}
</select>
```

Псевдонимы для выбираемых колонок должны полностью совпадать со значениями атрибутов *column* в мапперах. В данном примере используется элемент в фигурных скобках “#{id}” – это тот параметр, который мы передаем в данный запрос. Как это вызывается из Java-кода можно увидеть в листинге 5.

Листинг 5 – Пример вызова запроса из Java-кода

```
class User {
  private SqlSessionTemplate template;
  public User getUserById(String id) {
    return template.selectOne("IUserMapper.getUserById", id);
  }
}
```

SqlSessionTemplate является тем классом, через который происходит обращение к БД. В примере выше мы выбираем один элемент по id, который принимаем в качестве параметра метода.

Метод *selectOne* первым параметром принимает имя маппера и id запроса, который требуется выполнить, а вторым параметром уже сам параметр запроса.

Параметры, передаваемые в запросы, могут быть не только примитивными типами, но и сложными пользовательскими объектами. В таком случае для обращения к полям этих параметров следует обращаться через точку. Например, для параметра типа *User*, переданного в запрос, можем обратиться к его имени так: *#{user.name}*.

Для выполнения вставки, удаления, обновления данных используются теги *insert*, *delete*, *update*, соответственно. Кроме того, имеется возможность переиспользовать части SQL запросов, как представлено в листинге 6.

Листинг 6 – Пример переиспользования фрагмента SQL-запроса

```
<sql id="filtered">
  FROM users AS u
  INNER JOIN user_role AS ur
    ON u.id = ur.user_id
  INNER JOIN roles AS r
    ON ur.role_id = r.id
  WHERE u.id = #{id}
</sql>
-----
<select id="getUserById" resultMap="userMap">
  SELECT
    u.id AS u_id,
    u.name AS u_name,
    r.id AS r_id,
    r.name AS r_name
  <include refid="filtered"/>
</select>
```

Когда один и тот же фрагмент SQL-запроса используется в нескольких местах, уместно будет вынести этот фрагмент, как показано выше, и ссылаться на него в остальных местах. Это экономит место, а также способствует уменьшению объема работы, если данный фрагмент запроса в какой-то момент необходимо изменить. При таком подходе изменения будут вноситься только в одном месте, что экономит еще и время на внесение изменений.

Заключение

Представлен краткий обзор базовых возможностей MyBatis фреймворка и его базовой конфигурации. Также представлены примеры маппингов и запросов к БД.

Приведен пример вызова запросов из Java-кода.

Как видно, MyBatis не упраздняет наличие SQL-кода в проекте, однако он делает его написание и сопровождение гораздо легче.

Данная работа может служить отправной точкой в изучении фреймворка MyBatis.

ЛИТЕРАТУРА

1. Хабрахабр [Электронный ресурс] MyBatis как более быстрая альтернатива Hibernate. – Режим доступа: <https://habrahabr.ru/post/247885>. – Дата доступа: 27.09.2017.
2. Tutorialspoint [Электронный ресурс] MYBATIS Tutorial. – Режим доступа: <http://www.tutorialspoint.com/mybatis/>. – Дата доступа: 27.09.2017.
3. MyBatis [Электронный ресурс] Introduction. – Режим доступа: <http://www.mybatis.org/mybatis-3/>. – Дата доступа: 27.09.2017.

УДК 004.03

**СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИСПОЛЬЗОВАНИЯ ОБЛАЧНЫХ РЕШЕНИЙ
И СОБСТВЕННЫХ СЕРВЕРОВ ПРИ РАЗРАБОТКЕ ВЕБ ПРИЛОЖЕНИЙ****А.И. СТАТУТ***(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются плюсы и минусы двух подходов при разработке веб приложений. Приведены примеры технологий для разработки веб приложения в инфраструктуре Интернет-компании Amazon. Рассмотренные облачные технологии позволяют с легкостью развернуть и в будущем масштабировать веб приложения, однако обладают повышенной стоимостью и сложностью при миграции к другой компании либо на свой выделенный сервер.

Большое количество хостингов, которые доступны современным разработчикам и предприятиям, делает сложным первоначальный выбор, где разместить свое веб приложение. Выбор осложняет также различные типы решений, таких как выделенный сервер либо, так называемое облако, и каждый из этих типов предоставляет определенные преимущества. Важно, чтобы индивидуальные разработчики и компании подбирали решения для веб приложений исходя из своих потребностей и сильных сторон платформы хостинга.

Облачные вычисления

Облачный хостинг – это платформа виртуализированных серверов, где каждый облачный сервер является виртуальной машиной, которая работает на мощном выделенном сервере, и выделенный сервер поддерживает несколько облачных серверов [1]. Поскольку каждый виртуальный сервер является программной конструкцией, администратор может контролировать его, как и любое другое программное обеспечение. Виртуальный сервер можно создать за считанные секунды и при необходимости он может предоставить больше ресурсов, чем запрашивалось первоначально. Также виртуальный сервер может быть немедленно уничтожен, когда он больше не нужен.

Стоит также отметить, что облачные сервера могут быть настроены таким образом, чтобы обеспечить необходимые уровни производительности, безопасности и управления, аналогичные тем, какие предоставляются выделенными серверами, но вместо того, чтобы размещаться на физическом оборудовании, предназначенном исключительно для вас, они находятся в общей «виртуализированной» среде, которой управляет ваш провайдер облачных серверов. Благодаря этому, вы получаете выгоду от эффекта масштаба обмена оборудованием с другими клиентами. Поэтому облачные решения выгодны для малых и средних предприятий для оптимизации своих затрат, связанных с приобретением и управлением выделенной инфраструктурой. С облачными серверами клиент платит только за точный размер серверного пространства, которое он использует. Также присутствует возможность легкого масштабирования ресурсов, как и увеличение их, когда потребности клиента растут, так и уменьшения, чтобы избежать дополнительной платы за простой инфраструктуры, когда нагрузка минимальна.

Выделенные сервера

Выделенный сервер – это физический сервер, который приобретается либо сдается в аренду. Выделенные сервера отлично подходят для крупных предприятий и организациям, которым требуется исключительно высокий уровень безопасности данных. В отличие от облачных решений, все ресурсы выделенного сервера доступны клиенту.

Выделенный сервер может быть настроен в соответствии с точными требованиями клиента, который может выбрать процессор, оперативную память и хранилище, которые наилучшим образом соответствуют его потребностям. Выделенный сервер – это самый мощный вариант среди односерверного хостинга и единственным более мощным вариантом, чем выделенный сервер, является кластер выделенных серверов. Предприятия, которые используют приложения с большим количеством операций чтения и записей, например, базы данных либо обработчики большого количества данных, находят огромные преимущества в выделенных серверах.

Основные различия в облачных решениях и выделенных серверах

Если клиенту нужно разместить свое веб приложение в сети Интернет, то с помощью облачных серверов это можно сделать в считанные минуты. Если размещать веб приложение на выделенных серверах, то их покупка и настройка может занять несколько дней.

Если клиенту необходимо изменить размер выделенных ресурсов, таких как размер оперативной памяти, размер дискового пространства либо изменить процессор, то с облачными серверами это сделать можно также гораздо быстрее и в автоматическом режиме в отличие от выделенных серверов, где эта операция требует ручного процесса обновления.

Также стоит отметить, что автоматическое масштабирование доступно также только среди компаний, предоставляющих облачный хостинг.

Однако, если предприятию необходим полный контроль над своими приложениями и серверами, а также строгая безопасность и соответствие требованиям, то выделенные сервера будут лучшим выбором.

Не стоит забывать и про соотношения цены и производительности, в этом лучшее преимущество также за выделенными серверами.

Пример технологий для развертывания веб приложения в облачной платформе Amazon

Рассмотрим несколько технологий, с помощью которых можно легко управлять и масштабировать веб-приложения в облачной платформе компании Amazon [2].

AWSLambda – вычислительный сервис, который позволяет запускать код приложения без создания или настройки собственных серверов. AWSLambda выполняет код только в случае необходимости и автоматически масштабируется с нескольких запросов в день до тысяч в секунду. Клиент платит только за время вычислений, которые приложение непосредственно использует.

AmazonDynamoDB – это полностью управляемый сервис NoSQLбазы данных, обеспечивающий быструю и прогнозируемую производительность с плавной масштабируемостью. Клиент может использовать DynamoDB для создания таблиц базы данных, которые могут хранить и извлекать любой объем данных и обслуживать любой уровень трафика запросов. DynamoDB также автоматически распределяет данные и трафик для таблицы на достаточное количество серверов для обработки заданной клиентом емкости запроса и объема данных, сохраняя при этом постоянную и быструю работу.

AmazonS3 – это хранилище для статических данных, которое может быть использовано для хранения и получения любого количества данных в любое время.

Помимо вышеперечисленных технологий, Amazon предлагает большое количество облачных решений для нужд любого веб приложения.

Заключение

В действительности, облачные решения и выделенные сервера поддерживают любой тип приложений и сервисов, но каждый из них лучше всего подходит для конкретных сценариев и выбор правильной платформы может помочь компаниям снизить затраты, повысить гибкость и максимально использовать инфраструктуру за приемлемые деньги. Рассмотренные облачные технологии компании Amazon позволяют с легкостью развернуть и в будущем масштабировать веб приложения, однако обладают недостатками, такими как повышенная стоимость и сложность при миграции к другой компании либо на свой выделенный сервер.

ЛИТЕРАТУРА

1. Cloud computing: state-of-the-art and research challenges [Электронный ресурс]. – Режим доступа: <https://link.springer.com/content/pdf/10.1007%2Fs13174-010-0007-6.pdf>. – Дата доступа: 15.09.2017.
2. Документация AWS [Электронный ресурс]. – Режим доступа: <https://aws.amazon.com/documentation/>. – Дата доступа: 18.08.2017.

УДК 004.03

**СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ
И СРЕДСТВ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ****А.И. СТАТУТ***(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются методы разработки мобильных приложений, такие как разработка нативных приложений и гибридных. Также в статье произведен анализ преимуществ и недостатков разработки мобильных приложений с помощью различных типов программных библиотек.

Существует множество различных библиотек и других средств и инструментов разработки мобильных приложений. Каждый из них имеет свою особенность и предоставляет уникальные возможности, поэтому в данном исследовании поставлена задача – рассмотреть те инструменты, которые являются наиболее подходящими для реализации большинства мобильных приложений.

Операционные системы

Однако перед тем как говорить о видах мобильных приложений, стоит рассмотреть мобильные операционные системы (платформы).

На сегодняшний момент, по официальным данным, среди мобильных операционных систем в мире (и в частности в Беларуси) лидирует Android и iOS [1].

С учетом этих данных можно обосновать выбор операционных систем для разработки мобильного приложения, так как чем больше людей используют ту или иную платформу, тем выше вероятность того, что разрабатываемое приложение станет распространенным. Это утверждение не означает, что если разработано приложение под Android, то его сразу же загрузят тысячи людей, однако, если же приложение ориентированно на людей, проживающих в Республике Беларусь, и под операционную систему Windows Phone, то оно определенно обречено на провал, так как пользователей данной системы в нашей стране крайне мало.

Таким образом, при разработке приложения нужно учитывать предпочтения пользователей, на которых вы ориентируетесь. Для разработки мобильных приложений следует использовать наиболее распространенные операционные системы, какими в Республике Беларусь являются Android и iOS.

Типы мобильных приложений

Существуют два типа мобильных приложений: нативные и гибридные[2].

Нативные приложения разрабатываются инструментами, предоставленными Apple для iOS, Google для Android, Microsoft для Windows Phone и т.д., и стандартным языком для выбранной платформы, используется API, которое предоставляет данная платформа; интерфейс таких приложений организован и соблюдает стиль платформы.

Гибридные приложения чаще всего разрабатываются с помощью JavaScript, HTML5, CSS3. Инструменты, используемые для их написания, преобразуют веб-представление приложения в нативное, но в случае, если ведется разработка сложного приложения, уровни абстракции, используемые при преобразовании, чаще всего не позволяют использовать большинство, свойственных нативным приложениям, функциональных особенностей. Скорость работы таких приложений намного ниже нативных, а также они нуждаются в постоянном и непрерывном интернет-соединении. Перед тем как заняться разработкой, следует решить, для каких мобильных платформ будет реализовано приложение, сколько денежных средств будет вложено в проект.

Лучшим решением будет написание нативного приложения. Это обусловлено тем, что нативные приложения работают намного быстрее, а их интерфейс более стилизован. Но в этом случае затраты на разработчиков будут намного выше, так как потребуются команда специалистов с набором конкретных навыков. Если же требуется написать приложение, ориентированное на несколько платформ с наименьшими затратами и с удовлетворительным качеством, стоит задуматься о разработке гибридного приложения.

Анализ гибридных платформ

Если требуется разработать гибридное приложения, то, учитывая все недостатки данного вида приложений, необходимо отобрать несколько наиболее распространенных средств разработки и провести их сравнительный анализ. Рассмотрим следующие средства разработки:

1. IONIC.
2. AppceleratorTitanium.
3. ReactNative.
4. Cordova.

IONIC – это средство разработки приложений с открытым кодом для разработки мобильных гибридных приложений. Построен на основе AngularJS и Apache Cordova и обеспечивает разработку, используя такие веб-технологии как CSS, HTML5 и JavaScript. Приложения, созданные с помощью этих технологий распространяются через магазины приложений (Google Play или App Store и т.д.) [3].

Кроме функций, унаследованных от Apache, Cordova также предоставляет следующие возможности:

1. Выбор шаблона приложения.
2. Сборка и запуск в эмуляторе, на реальном устройстве, в браузере.
3. Генерация иконок и других элементов.

Поддерживаемые платформы: Android, iOS, Windows, BlackBerry.

Appcelerator Titanium – это фреймворк с открытым кодом, который позволяет создавать гибридные мобильные приложения с помощью JavaScript, разработанный Appcelerator.

Архитектурой этой библиотеки является кроссплатформенное API для доступа к нативным компонентам пользовательского интерфейса таким как: панели навигации, меню, диалоговым окнам, файловой система, сети, геолокации, акселерометр и карты. Весь исходный код приложения интерпретируется с использованием механизма JavaScript. Загрузка программы занимает больше времени, чем нативная программа, так как интерпретатор и все необходимые библиотеки должны быть загружены до интерпретации исходного кода. Некоторые разработчики сообщают: хотя работа с титаном дает быстрые результаты, что хорошо подходит при создании прототипов, однако есть вопросы, связанные с различиями в поведении с кроссплатформенным API, стабильностью и управлением памяти, что заставляло их переписывать свои приложения с помощью нативного кода.

Поддерживаемые платформы: Android, iOS, Windows, BlackBerry.

React Native – это платформа для создания гибридных приложений на основе JavaScript библиотеки React [4]. В отличие от других подобных платформ, которые запускают JavaScript приложения в веб-браузере, React Native компилирует в машинный код для соответствующей платформы. Это означает, что никакого снижения производительности приложения, созданного с помощью него не будет и приложение будет работать также как, как и приложения построенные с помощью нативных инструментов. Однако, кроме очевидных плюсов, платформа имеет множество недостатков: старые версии платформы не поддерживаются новыми, нет хорошей документации, всю необходимую информацию придется искать на форумах, трудоемкий процесс отладки приложений. Их можно объяснить тем, что платформа вышло относительно недавно и очень стремительно развивается.

Поддерживаемые платформы: Android, iOS.

Apache Cordova – одна из наиболее популярных платформ для разработки мобильных приложений. Она позволяет создавать гибридные мобильные приложения для мобильных устройств с помощью CSS3, HTML5, JavaScript. Приложения выполняются в обертках, которые являются платформо-ориентированными и опираются на стандартные API привязки для доступа к возможностям каждого из устройств, таких как датчики, данные, состояния сети и т.д. [5].

Поддерживаемые платформы: Apple iOS, Bada, BlackBerry, FirefoxOS, GoogleAndroid, LGwebOS, MicrosoftWindowsPhone, Tizen (SDK 2.x) и UbuntuTouch.

Если сравнить приведенные выше платформы, то среди них мы можем выделить IONIC и React Native, так как они являются наиболее быстрыми и продвинутыми по сравнению с Apache Cordova и Appcelerator Titanium. Ниже рассмотрим их сравнительный анализ на основе простейшего приложения с картой.

Сравнение платформы IONIC и ReactNative

Рассмотрим сравнение простейшего приложения, содержащего карту, разработанного на IONIC и React Native. Сравнение будет производиться по следующим критериям: сторонние модули, размер приложения, использование памяти.

1. Сторонние модули

Разрабатывая на React Native, мы можем использовать сторонние модули. В данном случае будем использовать нативное представление Google карт. В IONIC мы также можем использовать сторонние плагины, однако в случае с картами мы должны использовать WebView для их отображения. Недостатком данного подхода является более долгая загрузка и использование большего объема памяти. В нативном представлении карты мы можем использовать трехмерное отображение, масштабирование и компас.

2. Размер приложения

IONIC использует уже существующее WebView для отображения приложения, поэтому его размер достаточно небольшой. В то же время React Native создает мост между JavaScript и нативными компонентами, что сказывается на размере приложения. Одно и то же приложение в IONIC имеет размер всего 2.88 MB, в то время как в React Native – 8.22 MB. Разница довольно велика, хотя файлы приложения React Native можно сжать с помощью ProGuard. Однако в данном случае это сжатия не было выполнено.

3. Использование памяти

Количество памяти, которое использует приложение, очень важно, особенно для медленных устройств на Android. Поэтому там необходимо сохранить низкий уровень использования памяти так, чтобы приложение могло бесперебойно работать на устройствах. Если мы запустим команду `adb shell dumpsys meminfo`, то получим следующий результат: Ionic Framework использует ~ 84,714 kB памяти, в то время как React Native – ~ 58,585 kB. Таким образом, приложение работает лучше на React Native.

Подводя итог можно сказать, что наилучшим средством разработки гибридного мобильного приложения является React Native.

Заключение

Подводя итог, можно сказать, что выбор разрабатывать нативное либо гибридное мобильное приложение необходимо тщательно проанализировать, так как у каждого подхода есть свои преимущества и недостатки.

Если необходимо разработать приложение в сжатые сроки и на несколько операционных систем одновременно, но, возможно, в ущерб производительности, то лучше выбрать гибридные платформы.

Если проект имеет достаточный бюджет, приложение будет иметь сложные элементы интерфейса и важна производительность, то выбор за нативным методом.

Если сравнивать гибридные платформы, ReactNative является одним из лучших решений как компромисс между другими гибридными платформами и нативным способом разработки.

ЛИТЕРАТУРА

1. Восемь факторов, влияющих на стоимость разработки мобильного приложения [Электронный ресурс]. – Режим доступа: <https://link.springer.com/content/pdf/10.1007%2Fs13174-010-0007-6.pdf>. – Дата доступа: 10.09.2017.
2. Мобильные приложения: нативные vs html5 vs гибридные [Электронный ресурс]. – Режим доступа: <http://www.skywell.com.ua/blog/mobilnyie-prilozheniya-nativnyie-vs-html5-vs-gibridnyie/>. – Дата доступа: 07.08.2017.
3. Ionic framework. Обзор экосистемы [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/company/simpleweek/blog/254681/>. – Дата доступа: 07.08.2017.
4. React Native [Электронный ресурс]. – Режим доступа: <https://facebook.github.io/react-native/>. – Дата доступа: 07.09.2017.
5. Введение в Apache Cordova [Электронный ресурс]. – Режим доступа: <https://cordova.apache.org/docs/ru/latest/guide/overview/>. – Дата доступа: 17.09.2017.

УДК 004.021

**WEB-SCRAPING КАК ИНСТРУМЕНТ ИЗВЛЕЧЕНИЯ
ПОЛЕЗНЫХ ДАННЫХ ИЗ СЕТИ ИНТЕРНЕТ****Н.О. ШЕРШНЁВ****(Представлено: Ю.Н. КРАВЧЕНКО)**

Рассматриваются преимущества и недостатки получения данных из сети Интернет способом Web-scraping. Показаны основные способы анализа и обработки данных, их назначение и применение в повседневной жизни человека.

Информация играет чрезвычайно важную роль в жизни человека. Известно, что «Кто владеет информацией, тот владеет миром». В процессе развития современного общества все большие объемы несортированных данных хранятся в сети Интернет в открытом доступе. Поэтому при создании нового веб-ресурса перед разработчиком стоит выбор: использовать готовые данные из сети Интернет или прикладывать усилия для создания собственного контента. Это обусловило создание и разработку технологий, которые бы позволяли собирать полезную информацию в интернете, анализировать ее и предоставлять в структурированном виде конечному пользователю.

Веб-скрапингом называется такой подход к извлечению полезных данных веб-скрапингом, который подразумевает под собой создание специального программного обеспечения, позволяющего получать пользователю всю необходимую информацию с одного или нескольких интернет-ресурсов.

Существующие веб-скраперы имеют узконаправленную специализацию и зачастую создаются для конкретного веб-ресурса, что предполагает большие человеческие усилия для автоматизации процессов получения и дальнейшего преобразования информации к структурированному виду.

Веб-скраперы имеют широкое применение в разных сферах жизни человека.

При помощи веб-скраперов можно решить следующие задачи:

- мониторинг данных о погоде;
- сбор личных данных пользователей;
- поиск вакансий;
- работа с частными объявлениями по поиску жилья;
- отслеживание цен на товары в различных магазинах и т.д.

Среди готовых решений для скрапинга веб-сайтов стоит отметить следующие:

- веб-сервисы, которые работают через API (DiffBot, Embedly и др.);
- проекты с открытым кодом (Goose, Goutte, Morph, Scrapy и др.) [1].

Далее рассмотрим основные способы анализа и извлечения данных, использующие технологию веб-скрапинга:

1. **Copy-and-paste.** В некоторых случаях лучшим решением для получения данных из сети Интернет является простое копирование необходимой информации пользователем. Также этот способ полезен в том случае, когда веб-скрапер не может преодолеть защиту веб-сайта от машинной автоматизации.

2. **Text pattern matching.** Довольно простой, однако мощный подход к извлечению информации с веб-сервиса может быть основан на использовании регулярных выражений.

3. **HTTP programming.** Данный способ основывается на отправке http-запросов к удаленному веб-серверу, используя программирование сокетов.

4. **DOM parsing.** Программа, созданная на основе данного подхода, встраивается в полноценный веб-браузер, например Internet Explorer или Mozilla Firefox. Система управления браузером анализирует веб-страницы в DOM-дереве, на основе которых программа получает необходимую информацию.

5. **Computer vision web-page analysis.** Существуют попытки создания такого программного обеспечения, которое использует машинное обучение и компьютерное зрение, идентифицирует и извлекает информацию с веб-страниц.

6. **HTML parsing.** Многие веб-сайты содержат коллекции страниц, которые были сгенерированы автоматически из таких источников, как базы данных. Данные обычно кодируются в похожие страницы с помощью общих скриптов и шаблонов. В процессе интеллектуального анализа данных, программу, которая определяет такие шаблоны, извлекает их содержимое и переводит в понятную форму, называют оболочкой или wrapper [2].

7. **Web-scraping.** Существует программное обеспечение, которое может быть использовано для настройки веб-скрапинг решений. Такое программное обеспечение может автоматически распознать структуру веб-страницы, что освобождает от написания веб-скрапинг кода [2].

Заключение

В ходе проведенного исследования рассмотрены основные способы анализа и извлечения данных, использующие технологию веб-скрапинга:

- Web-scraping как инструмент извлечения полезных данных из сети Интернет;
- основные способы анализа и обработки данных;
- назначение и применение их в повседневной жизни человека.

ЛИТЕРАТУРА

1. Википедия [Электронный ресурс] Web-scraping. – Режим доступа: https://en.wikipedia.org/wiki/Web_scraping. – Дата доступа: 26.09.2017.
2. Habrahabr [Электронный ресурс] Web-scraping с помощью Python. – Режим доступа: <https://habrahabr.ru/post/280238/>. – Дата доступа: 26.09.2017.
3. Habrahabr [Электронный ресурс] Web-scraping при помощи Node.js. – Режим доступа: <https://habrahabr.ru/post/301426/>. – Дата доступа: 26.09.2017.

УДК 004.021

**МОБИЛЬНОЕ ПРИЛОЖЕНИЕ «МУЗЫКАНТЫ РОССИИ»
ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ АНДРОИД****Н.О. ШЕРШНЁВ****(Представлено: Ю.Н. КРАВЧЕНКО)**

Рассматриваются назначение и функциональные возможности разработанного автором мобильного приложения «Музыканты России» для операционной системы Андроид. Приложение бесплатное и распространяется посредством официального магазина приложений Google Play Market. В настоящее время идет процесс развития данного проекта в Беларуси и Казахстане.

Музыка является неотъемлемой частью жизни современного общества. Миллионы людей по всему миру увлекаются музыкой, для многих это стало смыслом жизни. Постоянно развивающаяся музыкальная индустрия обуславливает появление новых музыкантов и музыкальных групп. Однако путь к известности тернист, и не каждый начинающий добивается внимания к своему творчеству. Поэтому помощь в развитии и становлении юных дарований очень важна. На помощь таким людям приходит проект «Музыканты России», который был создан в марте 2015 года.

Описание проекта

Данный проект предназначен для облегчения коммуникации между музыкантами и участниками музыкальной индустрии, обмена полезной информацией, поиска единомышленников для совместного творчества. Проект является уникальным на территории СНГ, поэтому он нашел поддержку у множества спонсоров.

Благодаря проекту «Музыканты России» налажено сотрудничество со многими музыкальными партнерами по всей стране. Под музыкальными партнерами понимаются: концертные площадки и репетиционные базы, музыкальные лейблы и интернет-магазины, учебные базы, а также концертные агентства. Разработана собственная программа лояльности, которая подразумевает предоставление клиентам персональных бонусных карт со скидками на услуги музыкальных партнеров. Вся эта полезная информация предоставлялась пользователям посредством веб-сервиса. Идея зарекомендовала себя хорошо и получила множество положительных отзывов от пользователей, однако для развития проекта дальше было принято решение о создании мобильного приложения «Музыканты России» для операционной системы Андроид.

Первая версия приложения была запущена в августе 2016 года. На данном слайде представлен первоначальный дизайн приложения. Для улучшения качества предоставляемых услуг в марте 2017 года было принято решение о внесении изменений в приложение, добавление нового функционала, а также создание нового дизайна.

Мобильное приложение разрабатывалось для операционной системы Андроид, на языке программирования Java. Для удобства навигации по музыкальным партнерам использовались карты Google Maps. Серверная часть разрабатывалась при помощи PHP и MySQL. Все выбранные технологии использовались по требованию заказчика.

Мобильное приложение «Музыканты России» обладает следующими функциональными особенностями:

- возможность выполнять работу с информацией, поступающей в разрабатываемое приложение, что включает в себя обеспечение пользователя удобными средствами ввода, а также возможность редактирования, добавления и удаления информации;
- наличие визуальной среды для просмотра и обработки информации, важными характеристиками которых являются дизайн пользовательского интерфейса, интуитивность и удобство навигации;
- возможность получения информации о музыкальных магазинах, репетиционных базах и концертных площадках. Предоставление этой информации в виде списков и при помощи карты Google Map;
- возможность получение последней актуальной информации о предстоящих мероприятиях в мире музыки;
- возможность выставления оценки для музыкального партнера;
- возможность редактирования личной информации пользователя;
- возможность подачи частного объявления о поиске музыкантов, или музыкальных коллективов для дальнейшего сотрудничества.

Заключение

Рассмотрено мобильное приложение «Музыканты России» для операционной системы Андроид, его назначение и функциональные возможности.

Приложение бесплатное и распространяется посредством официального магазина приложений Google Play Market.

Общее количество скачиваний приложения составило 7 096, за последние два месяца приложение скачали 2 728 раз.

Приложение в дальнейшем планируется развивать. В настоящее время идет процесс развития данного проекта в Беларуси и Казахстане.

ЛИТЕРАТУРА

1. Википедия [Электронный ресурс] Web-scraping. – Режим доступа: https://en.wikipedia.org/wiki/Web_scraping. – Дата доступа: 26.09.2017.
2. Habrahabr [Электронный ресурс] Web-scraping с помощью Python. – Режим доступа: <https://habrahabr.ru/post/280238/>. – Дата доступа: 26.09.2017.
3. Habrahabr [Электронный ресурс] Web-scraping при помощи Node.js. – Режим доступа: <https://habrahabr.ru/post/301426/>. – Дата доступа: 26.09.2017.

УДК 004.77

ПОНЯТИЕ ТЕХНОЛОГИИ БЛОКЧЕЙН

К.С. ГОЛОВАН

(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Раскрываются понятия блокчейна, децентрализованной сети, разрешения консенсуса, частного и публичного блокчейна. Приводятся примеры современных блокчейн-решений. Блокчейн позволяет безопасно распространять и/или обрабатывать данные между несколькими лицами через недоверенную сеть. Наиболее интересным вариантом данных является возможность передачи информации, которая требует наличия третьей доверенной стороны.

Блокчейн – это журнал с фактами, реплицируемый на несколько компьютеров, объединенных в сеть равноправных узлов (P2P). Фактами может быть что угодно – от денежных операций и до подписания контента. Члены сети – анонимные лица, называемые узлами. Все коммуникации внутри сети используют криптографию, чтобы надежно идентифицировать отправителя и получателя. Когда узел стремится добавить факт в журнал, в сети формируется консенсус, чтобы определить, где этот факт должен появиться в журнале; этот консенсус называется блоком.

Децентрализованные сети с равноправными узлами появились задолго до блокчейна. К примеру, BitTorrent, только вместо обмена файлами в блокчейне обмениваются фактами.

P2P-сетям, как и прочим распределенным системам, приходится решать сложную проблему информатики – разрешение конфликтов, или согласование. Реляционные базы данных предлагают ссылочную целостность, но такой особенности нет в распределенной системе. Если два несовместимых факта прибывают в одно и то же время, система должна иметь правила для определения того, какой факт считать правильным.

В P2P сетях, два факта отправленные примерно в одно время могут прибыть в разном порядке в удаленные узлы. Тогда, каким образом всей сети согласовать, какой факт пришел первым?

Чтобы гарантировать целостность в P2P сети, вам нужен способ согласования порядка фактов. Вам нужна система консенсуса.

Алгоритмы консенсуса для распределенных систем это очень активное поле для исследований [1].

Задача распределенного консенсуса не специфична для блокчейнов и имеет хорошо проверенные решения для многих других распределенных систем (например, баз данных NoSQL). Но биткойн и другие блокчейны от предыдущих наработок отличаются условиями работы сети.

В обычных алгоритмах византийского консенсуса у узлов сети есть «личности», выражаемые через цифровые подписи или сходные криптопримитивы, а сам список узлов известен заранее или меняется редко, но предсказуемо. В биткойн-блокчейне все наоборот.

Участники сети не только заранее неизвестны, но и могут свободно подключаться или отключаться от сети. При этом блокчейн, являясь децентрализованной системой, имеет определенные свойства: устойчивость к цензуре (никто не может запретить майнить криптовалюту) и объективность (для определения текущей версии журнала транзакций не нужно доверие неким авторитетным источникам – корень доверия находится в самом блокчейне).

Из-за этого обычные алгоритмы византийского консенсуса для блокчейна не подходят. Поэтому было предложено множество различных алгоритмов, среди которых выделяются две основные категории: алгоритмы на основе доказательства работы (proof-of-work) и алгоритмы на основе подтверждения доли (proof-of-stake).

В случае доказательства работы хеш сообщения, объединенного со специальным полем (nonce), должен быть меньше определенного значения (или начинаться с определенного числа нулевых битов). Nonce не несет смысла для самого сообщения – это поле перебирается автором доказательства, пока не будет найдено подходящее значение. Название «доказательство работы» отражает тот факт, что для нахождения nonce надо совершить вычислительную работу, ожидаемое количество которой измеримо. Например, если нужно, чтобы первые 16 бит хеша равнялись нулю, в среднем нужно перебрать 65 536 значений nonce.

Доказательство работы похоже на цифровые подписи – оно обеспечивает целостность сообщения, так как вероятность того, что один и тот же nonce подойдет для различных сообщений, очень мала. Доказательства также легко проверяются – достаточно провести лишь одну операцию хеширования. В отличие от подписей, создание доказательства работы не требует знания секретов, но «потребляет» больше вычислительных ресурсов. Доказательства работы используются узлами биткойна для определения состояния системы. Актуальный журнал транзакций определяется как цепочка блоков с наибольшей сум-

марной сложностью доказательств работы. Майнеры, соответственно, должны искать блок поверх этой цепочки. Но, теоретически, никто не запрещает создавать новые блоки на основе какого-то старого блока (иногда случаются расщепления – форки – блокчейна, потому что два майнера находят новый блок практически одновременно). Однако намеренные форки невыгодны экономически, потому что блоки из побочных ветвей блокчейна никем не учитываются и не приносят их создателям никакой прибыли – одни затраты на нахождение доказательства работы.

Консенсус Накамото обеспечивает два ключевых требования к блокчейну, которые мы выделили ранее. Из-за того, что доказательство работы не привязывается к личностям майнеров (в отличие от цифровых подписей/сертификатов), обеспечивается устойчивость к цензуре. А за счет того, что доказательства работы быстро проверяются и не требуют для проверки ничего, кроме блокчейна, достигается эффективность протокола.

Консенсус Накамото устойчив к атакам на сеть блокчейна, причем его безопасность достаточно хорошо исследована теоретически (в отличие от более новых предложений для блокчейнов). В оригинальной статье отмечено, что сеть биткойна продолжит адекватно функционировать, даже если половина майнеров начнет злонамеренную активность. Если же среди майнеров появится злонамеренное большинство (так называемая атака 51%), они смогут игнорировать блоки от всех остальных майнеров, чтобы забрать себе всю награду в сети, или, например, мошенничать, производя повторную трату средств за счет намеренных форков блокчейна. Однако при этом отметим, даже если все майнеры в сети сговорятся, они не смогут обойти базовые механизмы безопасности биткойна – например, не смогут похитить биткойны пользователей.

У консенсуса Накамото есть свойство, которое многими воспринимается как большой недостаток – для обеспечения безопасности нужно «работать», то есть производить доказательства работы. Вычисления, которые выполняются в рамках PoW, не служат никакой полезной цели, и это архитектурная особенность. Оказывается, очень сложно придумать доказательство работы, которое бы выполняло общественно полезную роль. Поэтому может показаться, что ресурсы (прежде всего, электричество) на майнинг тратятся впустую (если не считать, что они тратятся на обеспечение безопасности).

Помимо этого, майнинг все же подвержен проблеме централизации. Более 70% хешрейта биткойна на данный момент находится в одной стране – Китае. Многие криптовалюты пытаются децентрализовать майнинг за счет использования доказательств работы, которые экономически невыгодно вычислять на специализированном оборудовании, однако с таким подходом возникает другая проблема – если сделать выгодным майнинг биткойнов с помощью домашних компьютеров, то арендовать оборудование для атаки (или создать для нее ботнет) станет существенно дешевле и проще.

Пытаясь решить эти проблемы, сообщество предлагает множество алгоритмов консенсуса, которые не требуют «работы». Самая популярная категория таких алгоритмов основана на доказательствах доли (proof-of-stake, PoS). Доказательство доли похоже на доказательство работы, только вместо совершения определенной работы автор показывает, что у него есть доля в системе (например, в виде ненулевого баланса криптовалюты).

Однако у доказательства доли есть досадный недостаток в сравнении с PoW: поскольку доказательства доли базируются не на реальном мире (вычислительная мощность), а на чем-то внутри самого блокчейна (баланс криптовалюты), задача построения надежного PoS-алгоритма оказывается нетривиальной [3].

Согласно последним исследованиям в области блокчейнов компании BitFury, выделяются следующие типы блокчейнов:

Открытый блокчейн (англ. publicblockchain) – блокчейн, в котором не существует ограничений на чтение данных блоков (при этом данные могут быть зашифрованы) и ограничений на отсылку транзакций для включения в блокчейн.

Закрытый блокчейн (англ. privateblockchain) – блокчейн, в котором прямой доступ к данным и отправка транзакций ограничен определенным узким кругом организаций.

Общедоступный (инклюзивный) блокчейн (англ. permissionlessblockchain) – блокчейн, в котором не существует ограничений на личность обработчиков транзакций (т.е. пользователей, которые могут создавать блоки транзакций).

Эксклюзивный блокчейн (англ. permissionedblockchain) – блокчейн, в котором обработка транзакций осуществляется определенным списком субъектов с установленными личностями.

Приватные блокчейны имеют определенные преимущества: *во-первых*, низкая стоимость транзакций, поскольку проверка их валидности проводится доверенными и высокопроизводительными узлами вместо десятков тысяч пользовательских устройств, как в случае с общедоступными сетями; *во-вторых*, блокчейн можно настроить таким образом, что показатель TPS (TPS – transactionspersecond) будет значительно большим, чем у общедоступных сетей (по крайней мере, в ближайшем будущем). Единственным ограничением в этом случае остается пропускная способность самого слабого узла в сети.

Еще одним преимуществом частных блокчейнов может являться больший контроль над системой со стороны компании. Суть в том, что частный блокчейн позволяет, например, быстро обновлять функциональность. Поэтому он привлекателен для учреждений, работающих с реестрами и системами учета, поскольку формирует контролируемую и прогнозируемую среду, по сравнению с общедоступными блокчейнами (о которых речь пойдет ниже).

Более того, создание блоков в частном блокчейне зачастую не требует «доказательства работы» (proof-of-work). В качестве примера можно привести протокол создания блоков, используемый в BitShares. Имеется установленное число обработчиков транзакций N , каждый из которых обладает парой ключей – секретным и открытым. Создатели блоков известны и определяются по цифровой подписи в заголовке.

Операторы формируют блоки по очереди через заданные временные интервалы. Порядок создания блоков или фиксирован, или перемешивается после полного цикла (N блоков). Если оператор не сумел сформировать блок в отведенное ему время, то он пропускает раунд. Если такое поведение стало результатом злоумышленников, то ситуация расследуется. Таким образом, если обработчики транзакций являются единственными потребителями данных блокчейна, можно построить надежный протокол создания блоков (например, немного усложнив приведенный выше алгоритм), который не будет использовать доказательство работы.

И хотя частные блокчейны могут и не использовать доказательство работы, этот протокол все-таки может быть подключен для повышения уровня защищенности, упрощения аудита и, как результат, повышения контроля над системой для конечных пользователей. По сути, доказательство работы переводит доверие к блокчейну из субъективного (доверие к системе эквивалентно доверию к контролирующей его организации) к объективному (доверие к системе вытекает из математических законов и гарантированно высокой экономической стоимостью атаки на систему, которая не зависит от личности атакующего).

Публичные или общедоступные блокчейны могут быть прочитаны любым пользователем, каждый из которых имеет право формировать транзакции. При этом операции защищаются механизмами криптографической верификации, такими как доказательство выполнения работы или подтверждение доли (proof-of-stake).

«Контролирует» публичный блокчейн сразу все сообщество участников сети – разработчики, пользователи, поставщики услуг, майнеры – которые обеспечивают целостность сети и удобство работы в ней. Эффективность работы сети достигается с помощью обновлений протокола, предотвращающих вредные изменения. Именно поэтому система позволяет создавать децентрализованные приложения с минимумом затрат на техническое обслуживание.

Также они предоставляют способ защиты пользователей приложения от разработчиков, ограничивая возможности последних. В приложениях на публичном блокчейне разработчик не может сам по себе изменять код или данные.

Помимо этого, публичные блокчейны обладают сетевыми эффектами. Первыми пользователями приложений, построенных на публичном блокчейне, зачастую являются пользователи других приложений на том же блокчейне, которые узнали о них благодаря эффекту взаимодействия программ. Например, мобильный кошелек, работающий на публичном блокчейне, может добавить функцию для взаимодействия этого приложения с другими распределенными приложениями на этом же блокчейне, значительно расширив свою пользовательскую базу.

Стоит отметить, что публичные блокчейны позволяют разрешить проблему передачи «товаров». Например, если пользователь A хочет продать домен пользователю B , то они сталкиваются с определенными трудностями. Если A передаст домен первым, то рискует не получить деньги, в ином случае B рискует не получить «товар». Для решения этой проблемы применяются посредники, взимающие проценты за проведение транзакции.

Однако если блокчейн имеет систему доменных имен и валюту, расходы сокращаются до нуля с использованием смарт-контракта. Первый отправляет программе домен, а второй – деньги. Проблем не возникает, поскольку программа является доверенной, так как действует в публичном блокчейне [2].

Заключение

Блокчейн позволяет безопасно распространять и/или обрабатывать данные между несколькими лицами через недоверенную сеть. Данными может быть что угодно, но наиболее интересным вариантом данных является возможность передачи информации, которая требует наличия третьей доверенной стороны. Примерами такой информации являются деньги (требуют участия банка), права на собственность (требуют участия нотариуса), договор на заем и т.д. В сущности, блокчейн устраняет необходимость в участии третьего доверенного лица.

С технической точки зрения блокчейн является новшеством, которое опирается на три понятия: P2P сети, асимметричная криптография и распределенный консенсус, основанный на решении математической задачи.

Блокчейн можно рассматривать как слабоинхронизированную базу данных, реплицируемую столько же раз, сколько узлов в сети, или как суперкомпьютер, образованный комплексом всех CPU/GPU входящих в него узлов.

Одно из главных преимуществ блокчейнов по сравнению с другими моделями распределенных баз данных – интеграция обработки информации, слежения за корректностью и безопасностью в единый протокол, минимизирующий влияние человеческого фактора. Из-за юридических и технических причин учреждения, в которых задействованы финансовые системы учета или реестры, в среднесрочной перспективе могут быть заинтересованы в использовании блокчейнов с ограниченным доступом к обработке транзакций.

Что касается публичных блокчейнов, то их преимущества (в частности, их прозрачность и открытость базовых технологий и протоколов) могут привести к тому, что технология заменит многие функции традиционных финансовых институтов, изменив принципы работы финансовой системы

ЛИТЕРАТУРА

1. Объяснение блокчейна для веб-разработчиков [Электронный ресурс] / Ч. 1 : Теория. – Франция, 2016. – Режим доступа: <https://marmelab.com/blog/2016/04/28/blockchain-for-web-developers-the-theory.html>. – Дата доступа: 30.09.2017.
2. Различия, достоинства, недостатки: публичные и приватные блокчейны [Электронный ресурс]. – Россия, 2016. – Режим доступа: <https://habrahabr.ru/company/bitfury/blog/324458/>. – Дата доступа: 30.09.2017.
3. «Алгоритмы консенсуса»: Подтверждение доли и доказательство работы [Электронный ресурс]. – Россия, 2016. – Режим доступа: <https://habrahabr.ru/company/bitfury/blog/327468/>. – Дата доступа: 30.09.2017.

УДК 004.9

ПРОГРЕССИВНЫЕ ВЕБ-ПРИЛОЖЕНИЯ

К.С. ГОЛОВАН

(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Раскрываются понятия нативных приложений, прогрессивных веб приложений, манифеста, ServiceWorkers. Приводится сравнение нативных и прогрессивных приложений.

Интерес к разработке приложений для мобильных операционных систем в сообществе разработчиков стабильно растет, количество инструментов и подходов позволяющих создавать приложения становится все больше. Примерно с 2005 года веб-технологии перешли от использования статических страниц на динамические, т.е. генерируемые сервером (PHP, ASP.NET) и клиентской стороной (AJAX), а также пришли к понятию респонсивный дизайн. Однако с выходом первой линейки телефонов iPhone данные технологии отошли на второй план, так как нативные приложения предлагали более быструю работу приложения и удобный интерфейс для пользователя. В середине 2010 года улучшения в HTML5, CSS3 и JavaScript наравне с развитием браузерных движков сделали так называемые гибридные приложения настоящей альтернативой нативным приложениям. Гибридные приложения «воспроизводят» нативный интерфейс и требуют установки из специального хранилища приложений. Комбинация скриптов, стилей и разметки дает возможность создавать интерфейсы без использования технологии Flash. Также гибридные приложения не имеют доступа по URL, поддерживают богатый пользовательский интерфейс и имеют доступ к операционной системе. В 2015 году компанией Google был представлен один из новых подходов к разработке веб-приложений, а именно прогрессивные веб-приложения (ProgressiveWebApps, прогрессивные веб-приложения).

Прогрессивные веб-приложения используют преимущества новых технологий, чтобы принести пользователю все самое лучшее из мобильных сайтов и нативных приложений. Они надежные, быстрые и привлекательные. Они доступны из безопасного источника и загружаются независимо от состояния сети.

PWA – это название группы приложений, которые используют стек Web технологий (JS + HTML + CSS) и позволяют соединить простоту использования Web сайта со специфичными для нативных приложений операционной системы UX и техническими возможностями.

Основное предназначение PWA увеличить конверсию, количество пользователей и удобство использования веб-приложений на мобильных устройствах.

ProgressiveWebApps является логическим продолжением AcceleratedMobilePages. Таким образом, если вы ранее создавали AMP приложения, то вам однозначно стоит обновить свое приложение к нормам PWA приложений. Если вы до этого ничего не слышали о AMP, то это не станет для вас проблемой во время изучения PWA.

PWA приложению необходимо соответствовать следующим требованиям:

- быть прогрессивным – работать с каждым пользователем вне зависимости от окружения, используя метод постепенного улучшения как основной принцип работы;
- адаптивным – подстраиваться под любое устройство: десктоп, смартфоны, планшеты или что либо другое;
- независимым от соединения, используя ServiceWorker приложение должно работать в оффлайн режиме при прерывании или отсутствии соединения;
- выглядеть нативно – приложение должно соответствовать привычным для пользователя способам взаимодействия и навигации;
- самообновляемым – приложение должно контролировать процесс автоматического обновления посредством ServiceWorker API;
- безопасным – посредством использования HTTPS предотвращать перехват и подмену данных;
- определяемым – посредством W3C манифеста и регистрации через ServiceWorker приложение идентифицируется как «приложение» поисковыми системами;
- удерживающим – используя технические возможности, мотивируем пользователя еще раз использовать приложение, например, посредством push уведомлений;
- легким в установке – позволяет «сохранить» приложение на устройстве пользователя посредством добавления PWA приложения в список установленных приложений без использования магазина приложений;
- легким в использовании – для начала использования приложения достаточно открыть URL. Установка приложения не обязательна.

Для создания прогрессивных веб-приложений используются следующие основные технологии: манифест и ServiceWorkers [1].

Манифест веб-приложения предоставляет информацию о приложении (такую как имя, авторство, иконку и описание) в формате JSON-файла. Цель манифеста – установить веб-приложение на домашний экран устройства, предоставляя пользователю более быстрый доступ и больше возможностей.

Манифест веб-приложения внедряется в HTML-страницу, с помощью тега ссылки в заголовке документа:

```
<link rel="manifest" href="/manifest.webmanifest">
```

Пример манифеста:

```
{
  "name": "HackerWeb",
  "short_name": "HackerWeb",
  "start_url": ".",
  "display": "standalone",
  "background_color": "#fff",
  "description": "A simply readable Hacker News app.",
  "icons": [{
    "src": "images/touch/homescreen48.png",
    "sizes": "48x48",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen72.png",
    "sizes": "72x72",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen96.png",
    "sizes": "96x96",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen144.png",
    "sizes": "144x144",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen168.png",
    "sizes": "168x168",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen192.png",
    "sizes": "192x192",
    "type": "image/png"
  }],
  "related_applications": [{
    "platform": "web"
  }, {
    "platform": "play",
    "url": "https://play.google.com/store/apps/details?id=cheeaun.hackerweb"
  }]
}
```

Манифест типового прогрессивного веб-приложения содержит параметры, которые определяют ожидаемый цвет фона для веб-приложения; описание того, что делает приложение; предпочтительный для разработчика режим отображения приложения; массив объектов изображений, которые могут быть

использованы как иконки приложения в различных контекстах, например в разных ОС; удобочитаемое имя, ориентацию [2].

На рисунке 1 представлен пример различного отображения приложения для разработчика.

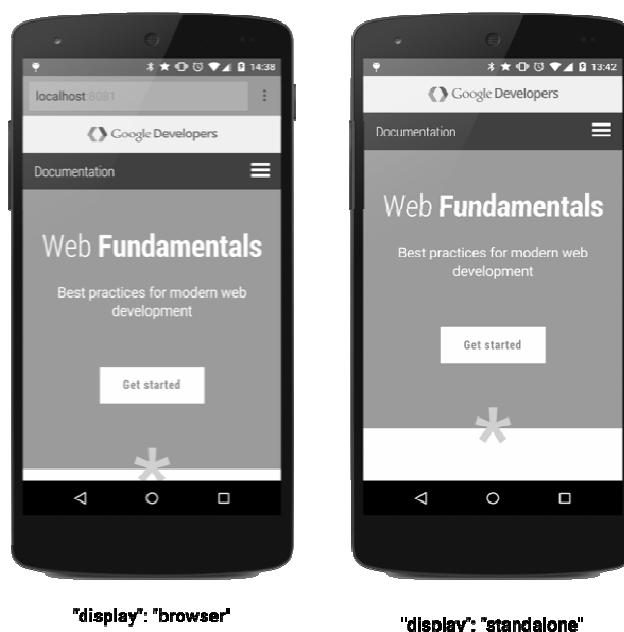


Рисунок 1. – Пример различного отображения прогрессивного веб-приложения при изменении параметра display в манифесте

Serviceworker – это событийно-управляемый worker, регистрируемый на уровне источника и пути. Он представляет собой JavaScript-файл, который может контролировать веб-страницу/сайт, с которым он ассоциируется, перехватывать и модифицировать запросы навигации и ресурсов, очень гибко кешировать ресурсы для того, чтобы предоставить вам полный контроль над тем, как приложение ведет себя в определенных ситуациях (например, когда сеть не доступна).

Serviceworker запускается в контексте worker'a, поэтому он не имеет доступа к DOM и работает в потоке отдельном от основного потока JavaScript, управляющего вашим приложением, а следовательно не блокирует его. Он призван быть полностью асинхронным; как следствие, синхронные API, такие как XHR и localStorage, в *serviceworker*'е использовать нельзя.

Из соображений безопасности *serviceworker*'ы работают только по HTTPS [3].

Serviceworkers также предназначены для использования для таких вещей, как:

- фоновая синхронизация данных;
- ответ на запросы от других источников;
- получение централизованного обновления для данных, сложных для расчетов, таких как геолокация или гироскоп, так что несколько станций может использовать одни и те же данные;
- компиляция на клиентской стороне и управление зависимостями для CoffeeScript, less, CJS/AMD модулей и т.д. для целей разработки;
- перехват фоновых сервисов;
- кастомная шаблонизация, основанная на определенных паттернах URL;
- улучшение производительности, к примеру, предварительная загрузка ресурсов, которые понадобятся пользователю в ближайшем будущем, как несколько последующих картинок в фотоальбоме.

В будущем *serviceworkers* будут способны на многие другие полезные вещи для web-платформ, приблизив их к жизнеспособности нативных приложений. Примечательно, что другие спецификации могут и будут использовать контекст *serviceworker*, например, для следующих целей:

- фоновой синхронизации: запускать *serviceworker*, даже когда ни одного пользователя нет на сайте, чтобы обновить кеш;
- реакции на пуш-сообщения: запускать *serviceworker* для отправки сообщений пользователям, чтобы оповестить их о новом доступном контенте;
- реакции на определенное время и дату;
- введение гео-ограничений.

Также некоторые PWA используют архитектурный подход, который называется AppShellModel. Для мгновенной загрузки сервис воркеры хранят базовый пользовательский интерфейс или «оболочку» (shell) приложения. Данная оболочка является статическим фреймом или макетом, в который контент загружается постепенно, при этом динамически позволяя пользователям взаимодействовать с приложением вне зависимости от качества соединения. Технически оболочка представляет собой код, который хранится в кэше браузера мобильного устройства.

Таким образом, стандартный процесс работы с прогрессивным веб-приложением заключается в следующем: пользователь, используя смартфон, открывает ссылку, полученную посредством любого приложения, после загрузки web страницы пользователь получает полноценное приложение, которое он может использовать.

Тем самым начать использовать новое приложение становится намного проще, так как не нужно заходить в магазин приложений и ждать пока установится необходимое пользователю приложение.

При необходимости пользователь может добавить приложение на рабочий стол посредством пункта опций в браузере «Addtohomescreen». Или приложение может предложить пользователю сделать это вместо него посредством Webapp-install-banner. После чего на главном экране пользователя будет создана иконка, предварительно указанная в манифесте приложения. Разработчик может управлять стилем браузера и вариантом отображения приложения после установки. Также ServiceWorker, идущий в комплекте с PWA приложением, будет отвечать за кеширование, оффлайн работу и обработку пуш сообщений.

Заключение

Прогрессивные веб-приложения являются инновационным подходом к разработке, так как включают в себе сильные стороны веб-сайтов (сами по себе ими и являются) и нативных мобильных приложений. То есть, с одной стороны, приложение имеет ссылочную структуру, небольшой размер, с другой – приложение ведет себя как настоящее нативное приложение с удобным и понятным интерфейсом. В основе PWA лежит ServiceWorker – посредник между устройством пользователя и сетью, который предоставляет контент вне зависимости от качества соединения и его состояния, при этом также предоставляет функции нативных приложений, такие как работа оффлайн, push-уведомления, экраны загрузки и другие.

На момент проведения исследований полная поддержка PWA обеспечивалась во всех браузерах, кроме Safari от Apple. С большей вероятностью данная технология не будет принята компанией Apple. Это может означать, что в ближайшем будущем может появиться новый подход в веб-разработке как конкурент прогрессивных приложений.

ЛИТЕРАТУРА

1. ProgressiveWebApps: WhoAmI[Электронный ресурс]. – Россия, 2017. – Режим доступа: <https://habrahabr.ru/post/303626/>. – Дата доступа: 03.10.2017.
2. Веб-документация MDN [Электронный ресурс] / Манифест веб-приложения. – США, 2017. – Режим доступа: <https://developer.mozilla.org/ru/docs/Web/Манифест>. – Дата доступа: 03.10.2017.
3. Веб-документация MDN [Электронный ресурс] / ServiceWorker API. – США, 2017. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/API/Service_Worker_API. – Дата доступа: 03.10.2017.

УДК 004.42+371.263

АНАЛИЗ СЛОЖНОСТИ ВОПРОСА В СИСТЕМЕ ДЛЯ ПРОВЕДЕНИЯ АДАПТИВНОГО ТЕСТИРОВАНИЯ

С.С. ВОЙТЕХОВИЧ*(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются основные принципы работы системы анализа сложности заданий в рамках системы для проведения адаптивного тестирования. Описано назначение подобной системы, возможный вариант реализации, достоинства и недостатки. Также показаны преимущества, получаемые в результате внедрения алгоритма анализа сложности в систему адаптивного тестирования.

Адаптивное тестирование – это способ контроля знаний, в котором совмещены достоинства оценки знаний с помощью компьютера и преподавательский подход контроля знаний. То есть во время проверки знаний в рамках устного опроса экзаменатор может отталкиваться от уровня знаний экзаменуемого, тем самым подобрав тот уровень сложности вопросов, на который экзаменуемый сможет ответить. Таким образом, под адаптивным тестированием понимается процесс проверки знаний, в рамках которого система способна регулировать сложность заданий в зависимости от уровня подготовленности тестируемого [1].

Для реализации такого процесса могут использоваться разные алгоритмы, но ключевой особенностью каждого из них является оперирование критерием сложности вопроса. Он необходим, так как система чаще всего не способна по формулировке вопроса определить его сложность, и ей требуется некоторое значение, от которого можно отталкиваться в процессе подбора нового набора заданий для тестируемого. Более того, возможность грамотно оперировать данным значением и отличает систему адаптивного тестирования от обычной системы компьютерного тестирования.

Способы определения сложности заданий

Потенциально самым простым способом получения значения сложности для системы является непосредственно составитель тестов. Тесты, так или иначе, должны быть кем-то созданы, и значение сложности для каждого конкретного вопроса может быть указано составителем заданий. Этой информации зачастую будет достаточно для правильной работы алгоритма адаптивного тестирования. Но данный подход имеет два недостатка:

- 1) необходимость составителю тестов анализировать каждый отдельный вопрос и грамотно оценивать его сложность;

- 2) возможная необъективность выбранной сложности для конкретного вопроса.

Говоря о второй проблеме, если в рамках устного опроса экзаменатор выберет неверную сложность вопроса для конкретного экзаменуемого, это создаст сложность только для одного человека. В случае же автоматизированной системы данная проблема может повлиять на прохождение неограниченного количества тестов.

Однако решение данной проблемы можно почерпнуть из того же примера с устным опросом. Экзаменатор, опираясь на ответы экзаменуемых, даже не осознавая сложности вопроса, со временем может ее оценить. То есть решением проблемы с необъективностью оценки сложности вопроса может стать возможность тестирующей системы анализировать результаты ответов на каждое конкретное задание. Потенциально, это же может стать вариантом и для решения первой проблемы. При создании набора тестовых заданий для каждого вопроса может быть указано некоторое значение сложности по умолчанию. В дальнейшем, с каждым новым ответом на конкретный вопрос его сложность будет меняться, и в результате будет получен набор заданий с градацией по сложности.

Таким образом, оценка сложности задания будет основываться не только на опыте и предпочтениях экзаменатора или составителя тестов, но и на реальных данных о прохождении тестов, что потенциально является более точным способом определения сложности заданий.

Вариант реализации системы анализа сложности

Система анализа сложности заданий может быть реализована по-разному. Можно учитывать разные параметры, такие как количество правильных/неправильных ответов, результаты тестируемого по прохождению других тестов по данному предмету (если система хранит такую информацию), время, суммарно затраченное на выполнение задания и другое.

Для примера можно рассмотреть систему, для которой показатель сложности вопроса определяется некоторым значением от 1 до 10, где 1 – самые легкие задания, 10 – самые сложные. При составлении теста некоторому вопросу был задан средний уровень сложности, то есть значение 5. После некоторого количества прохождений данного задания тестируемым имеем соотношение правильных ответов к неправильным, как 4 к 6-ти, то есть на данный вопрос было дано больше неправильных ответов, чем пра-

вильных. На основании этих данных легко определить, что сложность данного задания можно поднять до значения 6.

Потенциально, имея набор тестовых заданий с одинаково равным уровнем сложности, уже после первого его использования мы получаем градацию вопросов с двумя или тремя (в случае, если были показаны не все вопросы) уровнями сложности. И с каждым новым прогоном данного тестового набора количество градаций будет увеличиваться [2].

Рассмотренный вариант потенциально является самым простым способом реализации системы оценки сложности заданий на основании данных по ответам. Он не лишен недостатков как с концептуальной точки зрения, так и с точки зрения программной реализации.

С концептуальной точки зрения, по возможности, должны соблюдаться следующие требования:

- 1) тестируемые не должны знать о системе оценки сложности заданий;
- 2) тестируемые должны быть заинтересованы в получении наивысшей оценки.

Данные требования обусловлены тем, что, зная принцип работы системы, тестируемые могут напавленно пытаться контролировать предоставляемую выборку заданий, намеренно ошибаясь в некоторых вопросах, тем самым повышая их сложность и снижая вероятность их появления в будущем. Такой вариант весьма маловероятен, ибо в таком случае многие в результате тестирования могут получить плохую оценку, но возможен.

С точки зрения реализации данный алгоритм требует хорошо продуманного подхода к организации совместного доступа к данным. То есть при активном использовании системы тестирования, одно и то же задание одновременно может быть показано нескольким тестируемым, и нужно предусмотреть возможность попытки единовременного изменения сложности по результатам нескольких тестов.

Также может понадобиться реализация повторного использования заданий в рамках нескольких тестовых наборов с целью повторного использования информации о его сложности и получения большего количества данных для ее изменения. Это требование имеет и обратную сторону – может понадобиться предоставить возможность использования одного и того же задания с разной и независимой сложностью.

Заключение

Хотя вышеописанные недостатки и имеют место, существует малая вероятность того, что они могут создать большие проблемы для работы адаптивных алгоритмов в целом. Более того, работа адаптивного алгоритма тестирования и алгоритма подбора сложности для задания никак прямо не зависят и попросту используют результаты работы друг друга. Следовательно, система подбора сложности задания может разрабатываться как дополнительная функция системы адаптивного тестирования с возможностью отключения этой функции в случае необходимости. Аналогично, система анализа сложности может использоваться и без адаптивных алгоритмов, чтобы анализировать сложность вопроса для дальнейшего определения их «стоимости» при выставлении отметки или для предоставления дополнительной информации экзаменатору.

При своей достаточно несложной реализации система анализа сложности может упростить работу с системой адаптивного тестирования в целом. Более того, при необходимости, с помощью данной системы, можно полностью скрыть работу адаптивных алгоритмов и от составителя тестов, освобождая последнего от необходимости дополнительного изучения данного вопроса и указания сложности заданий. В итоге можно получить систему с понятным механизмом добавления заданий в базу данных и преимуществами адаптивных алгоритмов.

ЛИТЕРАТУРА

1. Словарь терминов [Электронный ресурс]. – Режим доступа: <http://www.profcareer.ru/lib/dicty.php>. – Дата доступа: 20.09.2017.
2. Трудность теста и тестовых заданий [Электронный ресурс]. – Режим доступа: <http://testolog.narod.ru/Theory7.html>. – Дата доступа: 21.09.2017.

УДК 378.146.004.4

ИСПОЛЬЗОВАНИЕ ЭЛЕКТРОННЫХ СИСТЕМ ОБУЧЕНИЯ ДЛЯ КОНТРОЛЯ ЗНАНИЙ

С.С. ВОЙТЕХОВИЧ*(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)*

Рассматриваются современные электронные системы контроля знаний, их характеристики и возможности использования в системе образования. Также подробно описан процесс применения систем на практике, их основные плюсы и минусы.

В век широкого распространения различных информационных и технических ресурсов все еще используются принципы контроля знаний, разработанные много десятилетий назад. Тем не менее в последнее время начали разрабатывать и новые системы, более подходящие для современной системы образования, которая приобретает все более массовый характер. Одним из самых распространенных нововведений является тестирование, которое позволило произвести переход с бумажных источников на электронные системы обучения.

Как правило, электронные системы обучения внедряются в учреждениях образования для организации дистанционного (или заочного) обучения, но в последнее время все чаще данную систему используют и для аудиторной проверки знаний. Данные системы имеют ряд очевидных плюсов: мобильность (переносимость), вариативность, простота проверки решений, беспристрастное выставление отметок.

Мобильность (переносимость)

В связи с тем, что большинство ресурсов, представляющих собой системы контроля знаний, позволяют проводить не только тестирование в компьютерных классах, во время занятий, но и внеаудиторное регулярное тестирование. Например, студенту или ученику создается аккаунт на ресурсах, подобных x-TLS [1], INDIGO [2], Let'stest [3] или УСАТИК (Универсальная Система Автоматизированного Тестирования и Контроля) [4; 5], и сообщаются учетные данные, необходимые для авторизации. Далее преподаватель выкладывает учебный материал, по которому будет проводиться тестирование, создает тест по этому материалу, а затем выставляет этот тест для выполнения и указывает дату и время, до которых данное задание необходимо выполнить, а также максимальное количество попыток. Кроме того, в некоторых системах можно с помощью календаря управлять временем не только сдачи теста, но и временем появления этих тестов у учеников. Таким образом, организовывается отложенное тестирование.

Возможность указания времени появления теста и срока его сдачи позволяет ученикам выполнять задания в наиболее удобной для них обстановке, в максимально подходящее время. Так как удаленные системы работают круглосуточно, пройти тест можно как ранним утром, так и поздним вечером. Кроме того, ученики, у которых возникли проблемы со здоровьем или другие личные затруднения, могут пройти тест на домашнем компьютере, или даже со своего мобильного телефона в тот момент, когда это будет им удобно. Помимо прочего, если ученик заранее знает, когда появятся новые тесты и к какому сроку их необходимо сдать, это делает его более организованным, учит правильно распределять свое время на учебу и личные дела, а также позволяет качественно и без спешки подготовиться к выполнению задания.

Возможность указания количества попыток прохождения теста есть не в каждой системе, но она полезна. Таким образом, учащийся для того, чтобы пересдать плохой тест, может потратить некоторое время на повторное изучение материала и сразу же приступить к очередной попытке прохождения. Во-первых, это позволяет отстающим ученикам набрать хотя бы минимальный положительный балл. Во-вторых, стимулирует учащихся к повторному прочтению нового и повторению ранее изученного материала. В-третьих, преподавателю в таком случае нет необходимости составлять дополнительные задания для отстающих, так как даже после неудачного прохождения тестов ученик должен иметь возможность реабилитироваться.

Кроме вышеперечисленных ресурсов есть также программы, рассчитанные на проведение тестирования в компьютерных классах. У таких программ либо есть сервер с заданиями и клиенты, в которых авторизуются ученики (связь между ними осуществляется посредством локальной сети), либо для каждого теста генерируется отдельный модуль, который копируется на компьютеры учеников (связи между модулями нет). К таким программам можно отнести Айрен [6], OpenTEST 2.0 [7] и MyTestXPro [8]. Все эти программы могут быть использованы только для аудиторного тестирования, но это не является большим минусом, так как эти системы имеют все достоинства автоматизированных систем тестирования, а выполнение контрольных и экзаменационных тестов в любом случае должно проходить в аудитории под контролем преподавателя.

Вариативность

Как правило, тесты содержат в себе набор типовых вопросов и составление нескольких вариантов заданий представляет собой очень трудоемкую задачу. С другой стороны, если всем ученикам дать одинаковый набор заданий, то повышается возможность списывания, использования шпаргалок, а также с каждой новой попыткой понижается эффективность прохождения тестов. Ведь если ученик знает, на какие вопросы он ответил верно, он может просто повторить свои ответы, не вдумываясь в содержание вопроса и не анализируя его. В то же время, если ученик знает, какие именно вопросы будут в тесте, он акцентирует свое внимание на учебном материале, связанном именно с этими вопросами, и игнорирует другие, порой не менее важные вопросы. Из-за этого зачастую и появляются небольшие пробелы в знаниях, которые могут потом иметь катастрофические последствия.

Решением данной проблемы является такое свойство автоматизированных систем проверки знаний, как вариативность. Преподаватель создает набор вопросов, охватывающий всю предметную область указанной темы, а система, в свою очередь, выбирает из данного набора некоторое конкретное количество заданий и предлагает их ученику в произвольном порядке для выполнения. Такой подход заставляет учеников учить весь теоретический материал, провоцирует к пониманию темы, а не заучиванию номеров правильных ответов, а также гарантирует адекватность полученной оценки реальным знаниям ученика.

Простота проверки решений

Как правило, довольно большой проблемой для преподавателя является проверка результатов тестирования. В случае если тест проходит небольшое количество человек, проверить все тесты вручную не составляет особого труда, но если группа состоит из 20–30 человек, а тест состоит хотя бы из 10–15 вопросов из разных вариантов заданий, проверка всех решений превращается в длительный и трудоемкий процесс, в котором велик процент ошибок со стороны преподавателя, так как со временем человек устает, и его внимание рассеивается. В случае использования автоматизированных систем контроля знаний правильность ответа на вопрос проверяется автоматически, а после прохождения теста результаты видит не только преподаватель, но и ученик. Благодаря системе студент может самостоятельно отслеживать тенденции изменения качества знаний по предмету и оперативно получать информацию о только что пройденном тесте. Системы работают автоматически, и сбоев в выявлении ошибок быть не может, в отличие от проверки вручную.

Беспристрастное выставление отметок

Довольно большой проблемой для преподавателей зачастую становится процесс выставления отметок. В случае устного опроса или письменного экзамена сложно оценить знания ученика, так как существует масса критериев выставления отметок, и всех их нужно придерживаться. Тесты являются отличным решением данной проблемы. Если у преподавателя существует достаточный набор тестовых заданий по данной теме и хорошо продумана система оценивания каждого вопроса, то остается лишь корректно распределить, за какой диапазон баллов результата какая отметка ставится, и система расставит их автоматически. Некоторые системы позволяют выставять различную «стоимость» вопросов, например: легкий вопрос – 1 балл; средний – 2 балла; сложный – 3 балла.

Система также решает проблему возможности влияния личных отношений между учениками и преподавателем на результат проверки знаний. Система не имеет никаких личностных качеств и не учитывает сторонние факторы во время выставления отметок.

Заключение

На основании вышеизложенного материала можно констатировать, что автоматизированные системы оценки знаний – удобное и эффективное средство для проведения как промежуточного, так и итогового контроля знаний. Их использование значительно упрощает и ускоряет работу преподавателя, в то же время стимулирует учеников к наиболее подробному и всестороннему изучению предложенного преподавателем материала, подталкивает их к дальнейшему развитию самодисциплины и самоорганизации.

Единственный минус данных систем – отсутствие возможности проверки заданий творческого характера, таких как рисунки, сочинения, пересказы и тому подобное, но для этого разработаны особые методики проверки, которые нельзя привести к какому-либо четкому своду правил. Если бы творческие задания оценивались системой (даже использующей нейронные сети), результат был бы непредсказуемым.

ЛИТЕРАТУРА

1. Автоматизированная система обучения и контроля знаний на основе расширенных тестовых заданий x-TLS [Электронный ресурс] / О программе для тестирования знаний и конструкторе тестов x-TLS. – Киев, 2016. – Режим доступа: <http://xtls.org.ua/about.php.html>. – Дата доступа: 27.12.2016.

2. INDIGO [Электронный ресурс] / Программа для создания тестов и онлайн тестирования. – Ставрополь, 2016. – Режим доступа: <http://indigotech.ru>. – Дата доступа: 27.12.2016.
3. Система тестирования Let'stest [Электронный ресурс] / Конструктор для создания тестов. Онлайн сервис для проверки знаний учащихся и аттестации работников. – Воронеж, 2016. – Режим доступа: <https://letstest.ru>. – Дата доступа: 27.12.2016.
4. Компьютерная газета A-Z [Электронный ресурс] / УСАТИК – система тестирования знаний. – Минск, 2016. – Режим доступа: <http://www.nestor.minsk.by/kg/2006/29/kg62917.html>. – Дата доступа: 28.12.2016.
5. УСАТИК [Электронный ресурс] / Описание пакета УСАТИК. – Ульяновск, 2001. – Режим доступа: <http://www.usatic.narod.ru>. – Дата доступа: 28.12.2016.
6. Айрен [Электронный ресурс] / Программа тестирования знаний. – М., 2009. – Режим доступа: <http://irenproject.ru>. – Дата доступа: 05.01.2017.
7. Программа тестирования [Электронный ресурс] / Компьютерное тестирование студентов OpenTEST 2. – Киев, 2013. – Режим доступа: <http://opentest.com.ua/kompyuternaya-programma-testirovaniya-znaniy-opentest-2>. – Дата доступа: 05.01.2017.
8. Компьютерное тестирование знаний и компьютерные педагогические тесты [Электронный ресурс] / MyTestXPro – система программ для создания и проведения компьютерного тестирования, сбора и анализа их результатов. – Москва, 2016. – Режим доступа: <http://mytest.klyaksa.net>. – Дата доступа: 05.01.2017.

УДК 004.021

ФРЕЙМВОРК SEMANTICUI

А.И. ПИРОВИЧ

(Представлено: Д.В. ПЯТКИН)

Рассматривается современный фреймворк SemanticUI для разработки веб-приложений. Показана его уникальность и особенности отличия от других фронт-енд фреймворков.

SemanticUI – современный фреймворк для фронт-енд разработки веб-приложений, основанный на LESS и jQuery; имеет изящный, утонченный и адаптивный дизайн, который обеспечивает гибкий пользовательский интерфейс.

Semantic UI предлагает несколько способов изменения элементов пользовательского интерфейса. Для крупных проектов, которые основаны на создании персонализированного визуального языка, ориентированного на бренд, темы сайта позволяют изменять базовые переменные, управляющие SemanticUI, а также определять альтернативный переопределяющий CSS. Темы сайтов переносимы между проектами и влияют на скомпилированный код фреймворка.

Для небольших проектов, проектов с быстрыми сроками упакованные темы отлично подходят для заимствований из других проектов с открытым исходным кодом в этой области.

Для любого типа проекта важным местом для начала настройки Semantic UI является файл `site.variables` – файл переменных, который наследует все другие файлы переменных.

Семантический интерфейс уникален по двум причинам: *во-первых*, фреймворк структурирован; он использует пять описательных категорий для определения повторно используемых компонентов пользовательского интерфейса. Организованные вместе на странице создают визуальное представление, которое мы считаем веб-сайтом.

- **UIElement** является базовым строительным блоком. Он может использоваться в одиночку либо в группах. Например, кнопка может быть независимой или помещаться в группу кнопок;

- **UICollection** – это группа различных элементов, которые являются зависимыми друг от друга. Например, веб форма может иметь кнопки, инпуты, иконки и т.д.;

- **UI View** представляет общую часть контента веб-сайта. Это может быть раздел новостей, секция комментариев;

- **UI Module** – модули, с интерактивной функциональностью на основе JavaScript. Примерами служат аккордеон, попап окна, поиск и т.д.;

- **UI Behavior** – компонент, который не может существовать независимо. Он используется для добавления функциональности в другие компоненты. Например, Form Validation предусматривает проверку для компонента Form.

Почти у каждого компонента есть типы, состояния и варианты использования. Например, для компонента `button` некоторые из типов включают в себя: стандартную кнопку, кнопку со значком, анимированную кнопку. Также кнопка может находиться в активном, отключенном или загруженном состоянии. Наконец, кнопка может отличаться по размеру и цвету и может быть отформатирована как базовая, социальная, переключающая и др. Эта система дает вам большую гибкость при внешнем виде компонента.

Semantic UI не только значим и хорошо структурирован с точки зрения именования его классов, но также именования, определения и описания его компонентов. Его структура гораздо более смысловая, чем в Bootstrap или Foundation.

Во-вторых, уникальность семантического интерфейса заключается в том, что она предоставляет некоторые особые функции и компоненты, не присутствующие в других фреймворках. Например, `Feed` и `Comment` в `UI Views components` или `Sidebar` и `Shape` в `UI Modules`. Кроме этого, при взаимодействии с компонентами SemanticUI имеется возможность отладки в режиме реального времени. Просто откройте свою веб-консоль, и вы будете видеть, что точно передают ваши компоненты, что они делают.

Еще одна сильная сторона Semantic UI заключается в том, что он использует минимальный и нейтральный стиль, оставляя настройки открытыми для вас. Он включает важные и полезные вещи, игнорируя дополнительные функции, которые вы, вероятно, никогда не будете использовать. Кроме того, компоненты фреймворка являются переносимыми и автономными, поэтому вы можете взять и использовать только те, которые вам нужны.

В фреймворке используются элементы `em` и `rem` для своих элементов, что делает его полностью отзывчивым и адаптивным для любого размера. Вам нужно только изменить базовый шрифт, и все остальные элементы будут соответствующим образом отрегулированы.

Заключение

Подводя итог, отметим, что Semantic UI хорошо документирован, веб-сайт содержит множество примеров для разных компонентов.

Как можно видеть, SemanticUI является новым и в некоторых аспектах уникальным фреймворком, который хорошо документирован на английском языке. Официальный веб-сайт содержит множество примеров для разных компонентов. Кроме того, в нем есть руководство по стилю с методами и указаниями о том, как писать код.

ЛИТЕРАТУРА

1. SemanticUI [Электронный ресурс]. – Режим доступа: <https://semantic-ui.com/>. – Дата доступа: 26.09.2017.
2. SemanticUIIntroduction [Электронный ресурс]. – Режим доступа: <http://learnsemantic.com/developing/customizing.html#the-future-of-themes>. – Дата доступа: 26.09.2017.
3. Semantic UItutorials [Электронный ресурс]. – Режим доступа: <https://github.com/Semantic-Org/Semantic-UI/wiki/Tutorials>. – Дата доступа: 26.09.2017.

УДК004.021

ТЕХНОЛОГИЯ FLEXBOX

А.И. ПИРОВИЧ

(Представлено: Д.В. ПЯТКИН)

Рассматривается новая CSS технология Flexbox, ее основные преимущества и актуальность. Это технология позволяет заменить float и табличную разметку. Она поставляется с умным набором значений по умолчанию, которые упрощают создавать сложное форматирование с относительно небольшим кодом.

Долгое время единственными надежными кросс-браузерными инструментами, доступными для создания макетов CSS, были такие вещи, как float и position. Это прекрасно, и они работают, но в некотором смысле они также довольно ограничены.

Следующие требования трудно, иногда невозможно, достичь с помощью этих инструментов удобным и гибким способом:

- вертикальное центрирование содержимое блока внутри его родителя;
- создание всех дочерних элементов контейнера занимает равное количество доступной ширины / высоты независимо от ширины и высоты;
- создание макета со столбцами, в котором они имели одинаковую высоту, даже если содержат различный объем контента.

Использование Flexbox значительно облегчает эту работу.

Создание всех детей контейнера занимает равное количество доступной ширины / высоты независимо от ширины и высоты.

Создание всех столбцов в макете с несколькими столбцами допускает одинаковую высоту, даже если они содержат различное количество контента.

Существует три версии спецификации, но только одна из них действительно имеет значение «display: flex» с отличной поддержкой браузера.

Основными преимуществами flexbox являются:

- все блоки очень легко делаются «резиновым», что уже следует из названия «flex». Элементы могут сжиматься и растягиваться по заданным правилам, занимая нужное пространство;
- выравнивание по вертикали и горизонтали, базовой линии текста работает шикарно;
- расположение элементов в html не имеет решающего значения. Его можно поменять в CSS;
- это особенно важно для некоторых аспектов responsive верстки;
- элементы могут автоматически выстраиваться в несколько строк/столбцов, занимая все предоставленное место;

- множество языков в мире используют написание справа налево rtl (right-to-left), в отличие от привычного нам ltr (left-to-right). Flexbox адаптирован для этого. В нем есть понятие начала и конца, а не права и лева. То есть в браузерах с локалью rtl все элементы будут автоматически расположены в реверсном порядке.

Макет Flexbox состоит из flex container, который содержит flex items. Flex container имеет главную и поперечную ось. Направление главной оси задает свойство flex-direction. А поперечная ось всегда будет перпендикулярна главной (рис. 1).

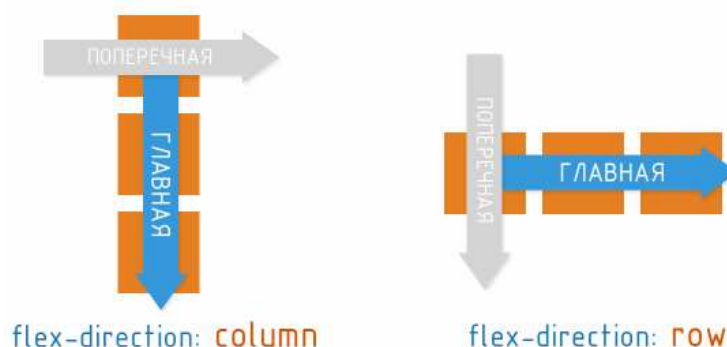


Рисунок 1. – Главная и поперечная ось Flexконтейнера

Flexbox включает в себя набор интеллектуальных значений по умолчанию, поэтому вы можете создать мощный макет с очень маленьким кодом. Пример представлен на рисунке 2.

```
.flex-container {
  display: flex;
}
```

Рисунок 2. – Пример использования flexbox

Эта одна строка кода выполняет все следующие действия:

- делает .flex-container как гибкий контейнер;
- рассматривает все дочерние элементы .flex-container как гибкие предметы;
- элементы Flex будут выложены в горизонтальной линии;
- элементы Flex будут размещены в исходном порядке;
- элементы Flex будут выложены с левой стороны гибкого контейнера;
- элементы Flex будут определяться на основе их обычных свойств width;
- если для всех элементов гибкости недостаточно места, им будет разрешено сжиматься горизонтально до тех пор, пока они не подойдут;
- если им нужно сжиматься, каждый элемент будет уменьшаться одинаково;
- элементы Flex будут растягиваться вертикально, чтобы соответствовать высоте самого высокого элемента.

Отсюда видно, насколько мощным может быть Flexbox без дополнительной работы.

Текущая спецификация Flexbox полностью поддерживается во всех современных браузерах, включая мобильные, IE11 и Edge (рис. 3). IE10 также поддерживает Flexbox, но использует tweenserspec (display:flexbox).

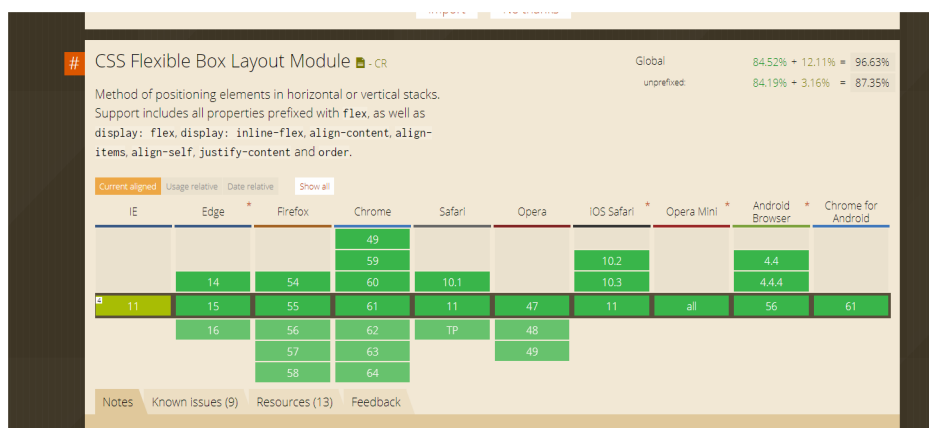


Рисунок 3. – Поддержка браузерами

Заключение

Flexbox – это новая технология, которая позволяет заменить float и табличную разметку. Она поставляется с умным набором значений по умолчанию, которые упрощают создавать сложное форматирование с относительно небольшим кодом. Flexbox стабильна, проста в использовании и хорошо поддерживаемая технология.

ЛИТЕРАТУРА

1. Whatisflexboxtechnology [Электронный ресурс]. – Режим доступа: <http://spaceninja.com/2015/08/24/what-is-flexbox/>. – Дата доступа: 26.09.2017.
2. Learnflexbox [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox. – Дата доступа: 26.09.2017.
3. Guidetoflexbox [Электронный ресурс]. – Режим доступа: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>. – Дата обращения: 26.09.2017.
4. Flexbox [Электронный ресурс]. – Режим доступа: <http://html5.by/blog/flexbox/>. – Дата доступа: 26.09.2017.
5. CanIUseFlexbox [Электронный ресурс]. – Режим доступа: <http://caniuse.com/>. – Дата доступа: 26.09.2017.

УДК 004.6

**МОДЕЛИРОВАНИЕ ДАННЫХ В NOSQL СУБД
НА ПРИМЕРЕ РАЗРАБОТАННОГО ВЕБ-ПРИЛОЖЕНИЯ
«ИНФОРМАЦИОННЫЙ АГРЕГАТОР БАНКОВ РЕСПУБЛИКИ БЕЛАРУСЬ»**

Р.П. ГУСЕВСКИЙ*(Представлено: Д.В. ПЯТКИН)*

Проанализированы отличия моделирования в NoSQL и SQLСУБД на примере разработанного веб-приложения «Информационный агрегатор банков Республики Беларусь», созданного с использованием одной из популярных СУБД MongoDB.

К настоящему времени человечеством накоплено поистине гигантское количество информации об объектах и явлениях. Но эта информация не лежит мертвым грузом, она хранится в электронном виде и используется в базах данных. Базы данных – это часть информационных систем – программно-аппаратных комплексов, осуществляющих хранение и обработку огромных информационных массивов.

База данных представляет собой определенным образом структурированную совокупность данных, совместно хранящихся и обрабатывающихся в соответствии с некоторыми правилами. Как правило, база данных моделирует некоторую предметную область или ее фрагмент. Программа, производящая манипуляции с информацией в базе данных, называется СУБД (система управления базами данных). Основными составляющими информационных систем, построенных на основе баз данных, являются файлы БД, СУБД и программное обеспечение (клиентские приложения), позволяющие пользователю манипулировать информацией и совершать необходимые для решения его задач действия. Базы данных как способ хранения больших объемов информации и эффективного манипулирования ею используются практически во всех областях человеческой деятельности.

Выбор NoSQLСУБДMongoDB

Для того чтобы хранить нужные данные, следует определиться с выбором БД, для реализации информационного агрегатора была выбрана база данных MongoDB, которая хорошо интегрируется с веб-приложениями. Это обусловлено тем, что MongoDB хранит данные в формате JSON. Формат JSON имеет хорошую поддержку в Javascript, поэтому работа с JSON хорошо отразится на производительности данного приложения.

MongoDB является NoSQL базой данных. NoSQL базы данных не работают с реляционными моделями. Существует много различных решений, каждое из которых работает по-своему и служит специфической цели. Эти безсхемные решения снимают ограничения с формирования сущностей и допускают хранения данных в виде ключ-значение.

NoSQL базы данных не используют общий формат запроса, такой как SQL в реляционных базах данных. Каждое NoSQL решение использует собственную систему запросов.

В MongoDB данные имеют гибкую схему хранения документов в одной коллекции. Это означает, что документы не должны иметь одинаковый набор полей или структуру. Общие поля в документах коллекции могут хранить различные типы данных [1].

Для того чтобы моделировать данные в MongoDB, нужно:

1. Комбинировать объекты в одном документе, если есть необходимость использовать их вместе.
2. Дублировать данные, так как время вычислений «дороже», чем место на диске.
3. Объединять данные во время записи, а не во время чтения.
4. Использовать сложные агрегации в схеме.
5. Оптимизировать схему под наиболее частые случаи.

Наглядная разница моделирования данных в NoSQL от реляционной базы данных

Чтобы более детально понять, в чем разница между моделированием в NoSQL базы данных и реляционной БД, рассмотрим следующий пример [2].

Предположим, нужно создать базу данных для хранения проектов и разработчиков, которые работают над этими проектами. Следует учесть, что каждый проект имеет уникальное имя и описание, и имеет много разработчиков.

Если использовать стандартную реляционную БД, то наша структура таблиц, примерно, имела вид, который представлен на рисунке 1.

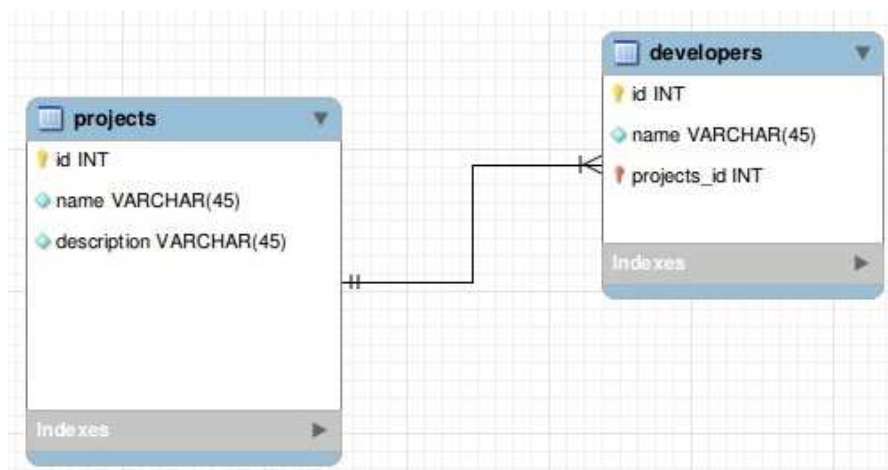


Рисунок 1. – Структура таблиц, рассматриваемая в примере

В MongoDB данная схема выглядела бы в виде структуры в формате JSON (листинг 1).

Листинг 1 – Пример схемы в MongoDB

```

{
  _id: PROJECT_ID
  name: NAME_OF_PROJECT,
  developers: [
    {
      name: 'DEVELOPER_NAME',
    },
    {
      name: 'DEVELOPER_NAME'
    }
  ]
}
  
```

Анализ и моделирование данных при разработке веб-приложения в MongoDB

При анализе функциональности веб-приложения выявлены следующие сущности:

1. Сущность *Обменный пункт* содержит данные о конкретном обменном пункте. Характеризуется наименованием, описанием, координатами: широты, долготы, городом, адресом.
2. Сущность *Отделение* описывает информацию о конкретном отделении банка. Характеризуется наименованием, описанием, координатами: широты, долготы, городом, адресом.
3. Сущность *Банкомат* содержит конкретную информацию о банкомате, принадлежащем конкретному банку. Характеризуется адресом, точной локацией, наименованием, временем работы, возможной валютой для снятия денег, координатами: широты, долготы.
4. Сущность *График Валют* содержит информацию для построения графиков валют. Характеризуется объектами валюты.
5. Сущность *Валюта* содержит описание валюты. Характеризуется наименованием валюты, ее кодом, массивом продажи и покупки данной валюты.
6. Сущность *Ссылки* содержит полезные ссылки банка. Характеризуется перечнем ссылок для банка.

Опираясь на рекомендации по моделированию данных в MongoDB, а также на результат краткого анализа функциональной части проекта смоделирована общая структура для хранения информации о банке (листинг 2).

Листинг 2 – Общая структура для хранения информации о банке

```
{
  nameBank: "ЗАО «БСББанк»",
  description: "Краткоеописаниебанка",
  shortName : "BSBBank",
  linkNews: "",
  linkBankCard: "",
  linkCredit: "",
  linkInternetBanking: "",
  linkInsurance: "",
  pathImg: "../img/banksLogo/bsbbank-logo.png",
  affiliate: [],
  exchangeOffice: [],
  bankomats: [],
  graphicCurrency: [],
  mainAddress: "",
  mainNumbers: "+375-17-306-20-40\n+375-29-306-20-40\n"+"+375-33-306-20-40",
  mainlinkAddress: "https://www.bsb.by/"
}
```

Из описания структуры видно, что каждое поле принимает значение различного типа, так как MongoDB хранит информацию в формате JSON [3], то в данном формате в качестве значений могут быть использованы:

1. Объект – это неупорядоченное множество пар ключ – значение, заключенное в фигурные скобки «{ }». Ключ описывается строкой, между ними значением стоит символ «:». Пары ключ – значение отделяются друг от друга запятыми.
2. Массив (одномерный) – это упорядоченное множество значений. Массив заключается в квадратные скобки «[]». Значения разделяются запятыми.
3. Число.
4. Литералы – true, false и null.
5. Строка – это упорядоченное множество из нуля и более символов, заключенное в двойные кавычки.

Исходя из того, что некоторые поля структуры (объекта) принимали в качестве значений массив, то данные таких полей представляли собой схожие объекты, которые нацелены на выполнение конкретного функционала.

Заключение

Авторами придумана при помощи длительного анализа общая структура, которая представляет собой объект, который описан выше (см. листинг 2). Данная структура хорошо подойдет для использования ее, как и для работы с графиками, так и для использования отображения местоположений банков, отделений и предоставления общей информации о банке. Выбор MongoDB в качестве СУБД приложения выбрана исходя из следующих плюсов:

- данная система хранения данных основана на принципе хранения документов в BSON (Binary JSON) формате, что делает ее хорошо интегрируемой с большей частью веб-приложений;
- более быстрое извлечение простых структур данных;
- хранение неструктурной информации;
- индекс по любому признаку;
- богатые запросы, репликация и высокая доступность [4].

ЛИТЕРАТУРА

1. METANIT.COM Сайт о программировании [Электронный ресурс]. – Режим доступа: <https://metanit.com/nosql/mongodb/1.1.php>. – Дата доступа: 17.09.2017.
2. Руководство по MongoDB. Моделирование данных [Электронный ресурс]. – Режим доступа: http://proselyte.net/tutorials/mongodb/data_modeling/. – Дата доступа: 19.09.2017.
3. Википедия : свободная энцикл. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/JSON/>. – Дата доступа: 19.09.2017.
4. W3IILatestwebdevelopmenttutorials [Электронный ресурс]. – Режим доступа: http://www.w3ii.com/ru/mongodb/mongodb_advantages.html. – Дата доступа: 22.09.2017.

УДК 004.021

**ВЫБОР ТЕХНОЛОГИЙ И РАЗРАБОТКА АДАПТИВНОГО ВЕБ-ПРИЛОЖЕНИЯ
«ИНФОРМАЦИОННЫЙ АГРЕГАТОР БАНКОВ РЕСПУБЛИКИ БЕЛАРУСЬ»****Р.П. ГУСЕВСКИЙ****(Представлено: Д.В. ПЯТКИН)**

Представлен практический способ создания интерфейса для веб-приложения «Информационный агрегатор банков Республики Беларусь». Целью работы явилось написание веб-приложения с удобным интерфейсом, который агрегирует данные с различных официальных банковских сайтов и предоставляет общие сведения по курсам валют в виде графиков, а также новостям и предоставляет ссылки на нужную информацию о банках, информация о местоположениях банкоматов, отделений и обменных пунктов.

Современному человеку представить жизнь без банка очень сложно. Сегодня через банки проходит практически вся наша финансовая жизнь: мы получаем зарплату на банковские карты, берем кредиты, ипотеку, платим через банки за образование детей, оплачиваем услуги ЖКХ, получаем и отправляем денежные переводы, обмениваем денежные средства. Из-за того, что данная тематика всегда будет актуальна, выбрана разработка веб-приложения «Информационный агрегатор банков Республики Беларусь». Немалую роль играют веб-приложения агрегаторы, где консолидируются сведения одновременно из нескольких источников. Они представлены в формате единого пользовательского интерфейса. Такого плана ресурсы крайне полезны при поиске наиболее выгодных условий предоставления услуг и быстрого получения всех необходимых данных. Приложения-агрегаторы – это правильный выбор, когда необходимо быстро найти искомое или сориентировать пользователя на целевые действия [1].

Разработка и выбор технологий для данного веб-приложения

Для реализации приложения были выбраны языки программирования: Javascript, Typescript, программная платформа NodeJS, фреймворк Angular 2, языком разметки веб-страниц является HTML, а также применялись каскадные таблицы стилей (CSS).

Node.js представляет среду выполнения кода на JavaScript, которая построена на основе движка JavaScriptChrome V8, который позволяет транслировать вызовы на языке JavaScript в машинный код. Node.js прежде всего предназначен для создания серверных приложений на языке JavaScript [2].

Для подключения MongoDB к NodeJS использовался фреймворк Express и система управления БД MongoDB для Windows Server 2008 R2 64-bit and later, with SSL support x64.

Express – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. Express позволяет иметь в своем распоряжении множество служебных методов HTTP и промежуточных обработчиков, что позволяет создать надежный API как можно быстрее и легче. Рассматриваемый фреймворк обладает хорошей производительностью за счет тонкого слоя фундаментальных функций веб-приложений, которые не мешают вам работать с функциями Node.js [3].

Чтобы установить связь между API и базой данных, хранящейся в MongoDB, нужно воспользоваться MongoDB Database Server, который устанавливается вместе с системой управления. Для хранения строки соединения с базой данных был создан отдельный модуль db.js (листинг 1).

Листинг 1 – модуль db.js

```
var MongoClient = require('mongodb').MongoClient;

var state = {
  db: null
};

exports.connect = function(url, done){
  if(state.db){
    return done();
  }
  MongoClient.connect(url, function (err, db){
    if(err){
      return done(err);
    }
  })
}
```

```
state.db = db;
  done();
})
};

exports.get = function(){
  return state.db;
};
```

Для лучшей работы с БД разделена логика на модель и контроллер. Создано два модуля: один модуль `banks.js`, который лежит в каталоге `controllers`, а другой модуль с аналогичным названием, который располагается в каталоге `models`.

В модели лежит основная логика для работы с базой данных в MongoDB, а в контроллере – логика для обработки запросов на сервер и получения информации пользователю, который использует данное API. Исходя из того, что модель и контроллер связан, будет предоставлен листинг с работой одного метода как в модели (листинг 2), так и в контроллере (листинг 3).

Листинг 2 – метод получения информации банка в модели

```
var db = require('../db');

exports.getBankByShortName = function(shortName,cb){
  db.get().collection('banks').findOne({ shortName: shortName}, function (err,doc) {
    cb(err,doc);
  });
};
```

Как описывалось ранее, логика контроллера напрямую зависит от логики модели, что описывается в примере, указанном ниже (листинг 3). Здесь установлена связь при помощи метода `require`, который принимает путь модуля и возвращает переменную-связь с данным унаследованным модулем. `Require` дает знать, что данный модуль наследует весь функционал из предыдущего модуля, обозначенный ключевым словом `exports`.

Листинг 3 – метод получения информации банка в контроллере

```
var Banks = require('../models/banks');
exports.getNameBanks = function(req,res){
  Banks.all(function (err, docs){
    if(err){
      console.log(err);
      return res.sendStatus(500);
    }
    docs = docs.map(function(bank){
      return {nameBank : bank.nameBank, shortName: bank.shortName ,pathImg: bank.pathImg};
    });
    res.send(docs);
  });
};
```

На клиентской части использовался фреймворк Angular 2. Он предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, шаблоны, маршрутизация и многое другое [4]. Одной из ключевых особенностей Angular является то, что он использует в качестве языка программирования TypeScript, что хорошо помогает в тестировании отдельных модулей и компонент приложения, так как этот язык, в отличие от Javascript, является строго типизированным.

Для того чтобы создать приложение на Angular 2, нужно установить Angular CLI, данный пакет поможет быстро развернуть и настроить проект. После разворачивания проекта необходимо создать модуль приложения, создать компоненты, разметку и стили для компонент и приступить к работе.

Весь дизайн сайта написан вручную, для достижения адаптивной верстки использован bootstrap 3.

При разработке дизайна клиентской части веб-приложения были учтены следующие *правила*:

1. Использование высококачественных изображений.
2. Существование каждого отдельного элемента на веб-странице должно быть чем-то обусловлено.
3. Изучение сайтов-аналогов.
4. Простота конструкции.
5. Подбор шрифта в сочетании с дизайном, шрифт одна из важнейших частей дизайна.
6. Больше графики, меньше текста.
7. Выделение важной информации.
8. Оптимизация сайта для удобства пользователя.
9. Умение продумывать и заполнять «воздушное пространство» на веб-сайте.
10. Краткая подача материала.

Главная страница с панелью навигацией показана на рисунке 1.

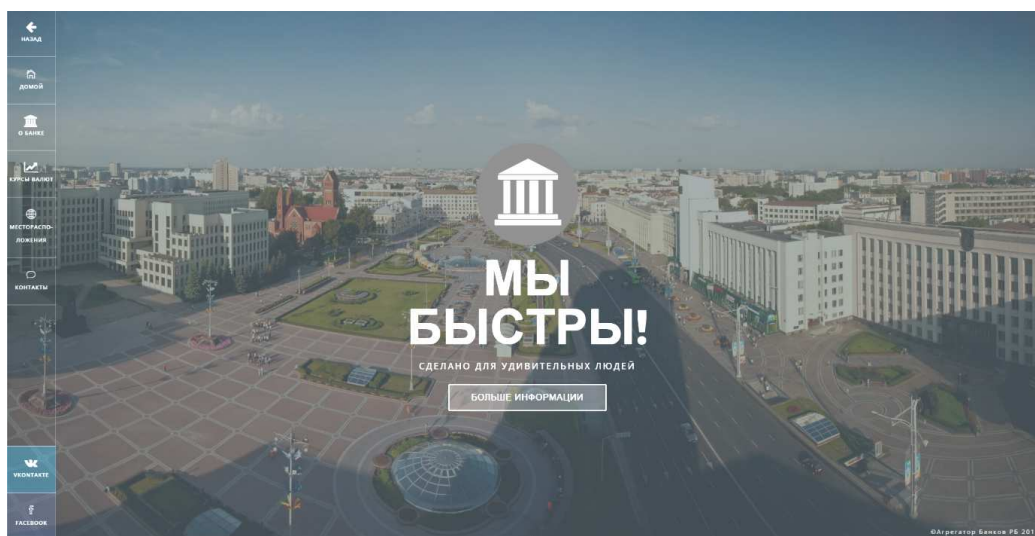


Рисунок 1. – Главная страница с панелью навигацией

Заключение

Авторами разработано веб-приложение с адаптивным дизайном, с серверной частью, которая была хорошо спроектирована и разделена на сущности контроллера и модели, что позволяет пользователю осуществить поиск нужной информации о банке в короткий промежуток времени. Надо всего лишь выбрать нужный банк и на панели навигации выбрать пункт, который нужен пользователю. На данный момент ведутся работы по увеличению функционала программы.

ЛИТЕРАТУРА

1. Мобильная газета бизнеса [Электронный ресурс]. – Режим доступа: <https://mobile.business-gazeta.ru/article/346729>. – Дата доступа: 19.09.2017.
2. METANIT.COM.NodeJS и MongoDB [Электронный ресурс]. – Режим доступа: <https://metanit.com/web/nodejs/6.1.php>. – Дата доступа: 23.09.2015.
3. Express [Электронный ресурс]. – Режим доступа: <http://expressjs.com/ru/>. – Дата доступа: 25.09.2015.
4. METANIT.COM.Руководство по Angular [Электронный ресурс]. – Режим доступа: <https://metanit.com/web/angular2>. – Дата доступа: 25.09.2017.

УДК 004.021

**ОПТИМИЗАЦИЯ АРХИТЕКТУРЫ БАЗ ДАННЫХ
С ЦЕЛЬЮ ПОВЫШЕНИЯ НАДЕЖНОСТИ И СКОРОСТИ РАБОТЫ****К.Х. ГРИГОРЯН****(Представлено: Д.В. ПЯТКИН)**

Рассматриваются технологии и приемы, для реализации отказоустойчивости серверной части приложения и увеличения отклика базы данных. Проанализированы общие приемы оптимизации работы с базами данных.

Довольно часто основной проблемой и упущением при работе с большими объемами данных является плохо организованная часть приложения, отвечающая за хранение данных. Современные технологии требуют повышенного внимания к back-end части приложения. А разработка back-end'a начинается с базы данных. Базы данных, их устройство и архитектура играют ключевую роль в работоспособности приложения и целостности данных пользователей.

Репликация

Репликация позволяет создать полный дубликат базы данных. Так, вместо одного сервера в разрабатываемом продукте будет несколько. Чаще всего используют схему master-slave:

1. **Master** – это основной сервер БД, куда поступают все данные. Все изменения в данных (добавление, обновление, удаление) должны происходить на этом сервере.

2. **Slave** – это вспомогательный сервер БД, который копирует все данные с мастера. С этого сервера следует читать данные. Таких серверов может быть несколько.

Репликация позволяет использовать два или больше одинаковых серверов вместо одного. Операций чтения (SELECT) данных часто намного больше, чем операций изменения данных (INSERT/UPDATE). Поэтому репликация позволяет разгрузить основной сервер за счет переноса операций чтения на Slave.

Для организации технологии репликации, в программном коде продукта необходимо объявить два соединения с базой данных. Одно – для мастера, одно – для Slave. Пример реализации предоставлен в листинге 1.

Листинг 1 – Несколько подключений в виде репликации.

```
<?
$master = mysql_connect('10.10.0.1', 'root');
$slave = mysql_connect('10.10.0.2', 'root');
// исполняемый код
$q = mysql_query('INSERT INTO users ...', $master);
// другая часть приложения
$q = mysql_query('SELECT * FROM users WHERE...', $slave);
```

Репликация обычно поддерживается самой СУБД (например, MariaDb) и настраивается независимо от приложения.

Следует отметить, что репликация сама по себе не очень удобный механизм масштабирования. Причина тому – рассинхронизация данных и задержки в копировании с Master на Slave. Однако это отличное средство для обеспечения отказоустойчивости. Вы всегда можете переключиться на слейв, если мастер ломается и наоборот. Чаще всего репликация используется совместно с шардингом, именно из соображений надежности.

Шардинг

Шардинг – это другая техника масштабирования работы с данными. Суть его в разделении базы данных на отдельные части так, чтобы каждую из них можно было вынести на отдельный сервер.

Есть несколько типов шардинга – вертикальный и горизонтальный.

Горизонтальный шардинг – это разделение одной таблицы на разные сервера. Это необходимо использовать для таблиц, которые хранят внушительные объемы информации и не умещаются на одном сервере. Разделение таблицы на куски делается по такому принципу:

- 1) на нескольких серверах создается одна и та же таблица (только структура, без данных);
- 2) в приложении выбирается условие, по которому будет определяться нужное соединение (например, четные на один сервер, а нечетные – на другой);
- 3) перед каждым обращением к таблице происходит выбор нужного соединения.

Предположим, что есть приложение, которое работает с высоконагруженной таблицей, хранящей фотографии пользователей. Также имеется два сервера (обычно они называются шардами) для этой таблицы. Возможный сценарий использования – для нечетных пользователей хранить и извлекать информацию, используя первый сервер, а для четных – второй. Таким образом, на каждом из серверов будет только часть всех данных о фотографиях пользователей. Пример представлен в листинге 2.

Листинг 2 – Несколько подключений, в стиле горизонтальный шардинг.

```
<?
$image_connections = [
    '1' => '10.10.0.1',
    '2' => '10.10.0.2',
];

$user_id = $_SESSION['user_id'];
// получение фотографий для пользователя $user_id
$connection_num = $user_id % 2 == 0 ? 1 : 2;
$connection =
mysql_connect($image_connections[$connection_num], 'root', '');
$q = mysql_query('SELECT * FROM images WHERE user_id = ' . intval($user_id), $connection);
```

Результат операции $\$user\_id \% 2$ будет остатком от деления на 2. То есть для четных чисел – 0, а для нечетных – 1.

Горизонтальный шардинг – это очень мощный инструмент масштабирования данных.

Вертикальный шардинг – это выделение таблицы или группы таблиц на отдельный сервер.

С целью применения такой разновидности шардинга, как вертикальный, необходимо использовать соответствующее соединение для работы с каждой таблицей.

Пример представлен в листинге 3.

Листинг 3 – Использование нескольких подключений в стиле вертикальный шардинг.

```
<?
$users_connection = mysql_connect('10.10.0.1', 'root', '');
$image_connection = mysql_connect('10.10.0.2', 'root', '');

$q = mysql_query('SELECT * FROM users WHERE ...', $users_connection);
$q = mysql_query('SELECT * FROM images WHERE...', $image_connection);

$q = mysql_query('SELECT * FROM albums WHERE...', $image_connection);
```

В отличие от репликации, мы используем разные соединения для любых операций, но с определенными таблицами.

Не следует применять технику шардинга ко всем таблицам. Правильный подход – это поэтапный процесс разделения растущих таблиц. Следует задумываться о горизонтальном шардинге, когда количество записей в одной таблице является заметным фактором на производительность работы приложения. Шардинг и репликация зачастую используются совместно. Пример такой реализации представлен в листинге 4.

Листинг 4 – Совместное использование шардинга и репликации

```
<?
$image_connections = [
    '1' => [
        'master' => '10.10.0.10',
        'slave' => '10.10.0.11',
    ],
    '2' => [
        'master' => '10.10.0.20',
        'slave' => '10.10.0.21',
    ],
];

$user_id = $_SESSION['user_id'];
```

```
// Читаем данные со слейвов
$connection_num = $user_id % 2 == 0 ? 1 : 2;
$connection = mysql_connect($image_connections[$connection_num]['slave'], 'root', '');
$q = mysql_query('SELECT * FROM image WHERE user_id = ' . intval($user_id), $connection);

//исполняемый код...
//Изменение данных происходит на мастерах
$image_id = 7;
$connection_num = $user_id % 2 == 0 ? 1 : 2;
$connection = mysql_connect($image_connections[$connection_num]['master'], 'root', '');
$q = mysql_query('UPDATE images SET views = views + 1 WHERE image_id = ' . intval($image_id),
$connection);
'2' => '10.10.0.2',
```

Такая схема часто используется не только для масштабирования, но и для обеспечения отказоустойчивости. Так, при выходе из строя одного из серверов шарда, всегда будет запасной.

Заключение

Итогом данной работы можно выделить рассмотренные некоторые возможности для грамотной реорганизации баз данных с целью улучшения оптимизации и увеличения отказоустойчивости.

ЛИТЕРАТУРА

1. Semantic UI [Электронный ресурс]. – Режим доступа: <https://semantic-ui.com/>. – Дата доступа: 26.09.2017.
2. Semantic UI Introduction [Электронный ресурс]. – Режим доступа: <http://learnsemantic.com/developing/customizing.html#the-future-of-themes>. – Дата доступа: 26.09.2017.
3. Semantic UI tutorials [Электронный ресурс]. – Режим доступа: <https://github.com/Semantic-Org/Semantic-UI/wiki/Tutorials>. – Дата доступа: 26.09.2017.

УДК 004.021

АВТОМАТИЧЕСКОЕ ПОЛУЧЕНИЕ СЕРТИФИКАТОВ ДЛЯ ОРГАНИЗАЦИИ ПРОТОКОЛОВ HTTPS

К.Х. ГРИГОРЯН*(Представлено: Д.В. ПЯТКИН)*

Рассматривается автоматическое получение сертификатов для организации протоколов https. Показан способ автоматизации процесса получения сертификатов для доменного имени. Важным аспектом процесса перевода сервера с http на https является фактор серверного окружения. В частности, подразумевается, что если на сервере установлено по apache2, то необходима перезагрузка. Текущие технологии позволяют обеспечить надежную защиту данных, в частности при использовании https.

Конфиденциальность передаваемых данных в сети является важным аспектом деятельности банковских систем или веб-приложений, хранящих персональные данные пользователя.

Утечка таких данных может нанести значительный вред как отдельным лицам, так и крупным организациям.

Для сохранения персональных данных выделено и разработано ряд мер и технологий, одной из которых является *https* протокол.

Lets Encrypt

Есть целый набор способов получить ssl-сертификат, но стоит выделить сервис Lets Encrypt, который отличает надежность и скорость получения сертификатов (за счет упразднения лишних шагов и возможности автоматизировать этот процесс), а также ряд программных средств, которые облегчают владельцу домена взаимодействие с этим сервисом.

Для автоматической выдачи сертификата конечному доменному имени используется **протокол аутентификации класса «challenge-response»** (вызов-ответ, вызов-отклик) под названием Automated Certificate Management Environment (ACME). В этом протоколе к веб-серверу, запросившему подписание сертификата, производится серия запросов для подтверждения факта владения доменом (domain validation).

Для получения запросов клиент ACME настраивает специальный TLS сервер, который опрашивается сервером ACME с применением Server Name Indication (Domain Validation using Server Name Indication, DVSNi).

С целью настройки системы обновления сертификатов на сервере, имеется ряд модулей для получения ssl сертификатов. Также важным аспектом используемого ПО является автоматическое привязывание сертификатов и перевод клиентов на 443 порт, для поддержания https.

Важным аспектом процесса перевода сервера с http на https является фактор серверного окружения. В частности, подразумевается, что если на сервере установлено по apache2, то необходима перезагрузка.

В случае работы с nginx достаточно будет использовать мягкую перезагрузку. Это приводит к мысли о том, что необходимо найти ПО для управления apache2/nginx.

Следовательно, после получения сертификата и добавления его в конфигурацию сайта необходимо автоматически перезагрузить сервер.

После этого сервер будет использовать обновленные конфигурации.

За короткое время клиент получает полностью работоспособный сайт с протоколом https.

Процесс верификации можно разделить на несколько стадий:

Создание открытого ключа по адресу domain/.well-known/acme-challenge/{challenge-token} – этот процесс необходим для подтверждения прав на домен.

Затем необходимо сделать **запрос с предоставлением данного файла сервису**.

После удачной проверки начинается **процесс получения сертификата**.

После обмена запросами имеем сертификаты в файлах fullchain.pem, cert.pem, chain.pem.

Сертификаты, как правило, выдаются на 2–3 месяца. Это необходимо учесть и добавить настройку для cron.

В основной файл конфигурации cron(/etc/crontab) необходимо добавить строку, представленную в листинге 1.

Листинг 1 – Листинг cron для Lets Encrypt

```
1: 0 0 0 */4 * /usr/local/bin/php /pathtoscript/LetsEncrypt.php
```

Таким образом, мы задали, что раз в три месяца будет обновляться сертификат.

В заключение можно сделать следующие **выводы**:

- текущие технологии позволяют обеспечить надежную защиту данных, в частности при использовании https;

- если раньше получение сертификата было сложной задачей, то при помощи сервиса Lets Encrypt эта задача перестала вызывать какие-либо проблемы у владельцев web-ресурсов.

ЛИТЕРАТУРА

1. METANIT.COM Сайт о программировании [Электронный ресурс]. – Режим доступа: <https://metanit.com/nosql/mongodb/1.1.php>. – Дата доступа: 18.09.2017.
2. Руководство по MongoDB. Моделирование данных [Электронный ресурс]. – Режим доступа: http://proselyte.net/tutorials/mongodb/data_modeling/. – Дата доступа: 17.09.2017.
3. Википедия : свободная энцикл. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/JSON/>. – Дата доступа: 21.09.2017.
4. W3I Latest web development tutorials [Электронный ресурс]. – Режим доступа: http://www.w3ii.com/ru/mongodb/mongodb_advantages.html. – Дата доступа: 23.09.2017.

УДК 004.05

ПОСТРОЕНИЕ ВЕБ-ПРИЛОЖЕНИЙ С ПОМОЩЬЮ ТРЕХСЛОЙНОЙ АРХИТЕКТУРЫ

И.В. МИСЕВИЧ*(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)*

Рассматривается трехслойная архитектура, при помощи которой ведется успешное построение веб-приложений. Приведен пример многослойной архитектуры – трехслойная архитектура построения приложений, а также приведены основные правила и рекомендации по организации и построению архитектуры приложения.

Многослойная архитектура является одной из архитектурных парадигм разработки программного обеспечения, при которой функциональные области приложения разделяются на группы.

Всю функциональность приложения, так или иначе, можно разделить на группы в зависимости от того, какая задача выполняется этой функциональностью, что приложение посредством ее делает: общается с пользователем или осуществляет проверку данных, или манипулирует данными из базы, или описывает сущности, или описывает логику.

Данная работа дает возможность получить представление о трехслойной архитектуре, понятии архитектурного слоя, правилах определения слоев приложения.

Трехуровневая (трехслойная) архитектура – архитектурная модель программного комплекса, предполагающая наличие в нем трех компонентов: клиента, сервера приложений, к которому подключено клиентское приложение, и сервера баз данных, с которым работает сервер приложений.

Для себя можно представить структуру приложения, построенного по принципу многослойной архитектуры, как пирамиду, в которой каждый следующий уровень расположен поверх предыдущего. Особенностью такой пирамиды является то, что называют перевернутой пирамидой повторного использования, когда каждый слой, расположенный выше, обладает информацией о том, как обратиться к слою, расположенному непосредственно под ним, и ни в коем случае не наоборот.

Рассмотрим классическую 3-слойную модель (UIL – BLL – DAL), но даже в ней могут присутствовать дополнительные прослойки.

Слой – это способ структуризации информации, находящейся в понятии архитектуры и указывающий, какой именно аспект выделяется и рассматривается.

В самом вершине пирамиды находится слой пользовательского интерфейса (UIL – User Interface Layer). Какой технологией он будет представлен – зависит от типа приложения. Это могут быть элементы Swing JSP технологии JavaEE и многое другое. Основная задача этого слоя – предоставлять эргономичный интерфейс пользователю в соответствии с функционалом, описанном в техническом задании. Этот слой просто должен быть красивым, удобным и интуитивно понятным. При этом он должен знать, каким образом взаимодействует со слоем, находящимся на уровень ниже, как он передает туда данные и как оттуда принимает данные для отображения.

Что же касается логики слоя, находящегося под ним, UIL об этом ничего не знает, а именно какие алгоритмы выполняются в тех методах, которые он вызывает по нажатию кнопки или по изменению текста в поле ввода, или при выборе пункта из списка.

Ниже находится слой бизнес-логики. BLL – Business Logic Layer: классы, описывающие сущности и основную логику работы приложения, основные алгоритмы работы приложения, вычисления, обработка. Например, если мы пишем сетевые «крестики-нолики», то на этом слое могут быть описаны классы, описывающие игровое поле, выполненный ход, отправку данных по сети и так далее.

Ниже находится слой доступа к данным (DAL – Data Access Layer). Это работа с источником данных – базами данных разных поставщиков MSSQL, MySql, Oracle, или XML файлами, или бинарными данными, или текстовыми файлами, всем тем, в чем можно данные хранить постоянно. Здесь также описана логика, но направленная именно на взаимодействие с источником и передачу полученных результатов на уровень выше.

Могут быть еще и другие слои, но какие и в какой последовательности они расположены в пирамиде архитектуры зависит от приложения.

Главное, что нужно знать – это набор следующих правил:

- каждый слой занимается конкретной задачей;
- логика разных слоев не повторяется и не пересекается;
- способ обращения к нижестоящему слою четко определен;
- способ поставки информации вышестоящему слою четко определен;

- слои слабо связаны;
- слои расположены вертикально, хотя есть сквозная функциональность, которая может пронизывать пирамиду сверху вниз;
- слои могут размещаться физически на одном компьютере (в пределах одного уровня), а могут быть на разных машинах, например, в распределенных приложениях;
- логика разных слоев инкапсулирована, соответственно, разным слоям не нужно делать никаких предположений о том, как реализован код других слоев приложения;
- возможность использования в других сценариях (за счет слабого связывания и четко определенной задачи).

По сравнению с двухзвенной клиент-серверной архитектурой или файл-серверной архитектурой трехуровневая архитектура обеспечивает, как правило, большую масштабируемость за счет горизонтальной масштабируемости сервера приложений и мультиплексирования соединений, большую конфигурируемость за счет изолированности уровней друг от друга.

Заключение

Представляемая работа дает возможность получить представление о трехслойной архитектуре, понятии архитектурного слоя, правилах определения слоев приложения

ЛИТЕРАТУРА

1. Трехуровневая архитектура : словари и энциклопедии Академик [Электронный ресурс]. – Режим доступа: <http://dic.academic.ru/dic.nsf/ruwiki/198141>. – Дата доступа: 23.09.2017.
2. Разговор о MVC и архитектуре веб-приложений [Электронный ресурс] // Хабрахабр. – Режим доступа: <https://habrahabr.ru/post/283096>. – Дата доступа: 25.09.2017.
3. Электронный университет ВГУ [Электронный ресурс]. Трехзвенная архитектура с сервером приложений. – Режим доступа: <https://edu.vsu.ru/mod/book/view.php?id=12520&chapterid=237>. – Дата доступа: 25.09.2017.

УДК 004.05

СРАВНЕНИЕ ФАБРИК. ФАБРИЧНЫЙ МЕТОД, АБСТРАКТНАЯ ФАБРИКА И ДРУГИЕ

И.В. МИСЕВИЧ*(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)*

Рассматриваются вариации порождающих шаблонов проектирования – фабрики, их особенности, отличия. Представлен краткий обзор шаблонов проектирования – фабрики. Приведены примеры реализации данных шаблонов.

Для упрощения проектирования и поддержки программ часто используют типовые унифицированные решения – паттерны. За удобное и безопасное создание новых объектов или даже целых семейств объектов отвечают порождающие паттерны, а именно фабрики. Во многих книгах и источниках определения «фабрик» дается авторами по-своему. Это создает большую путаницу при чтении различных материалов. В данной статье рассмотрим вариации фабрик, что бы понять разницу между ними.

Основной раздел

Фабрика – это общая концепция проектирования функций, методов и классов, когда какая-то одна часть программы отвечает за создание других частей программы.

В общей концепции имеется в виду:

- функция или метод, создающий все объекты программы;
- класс, создающий пользователей системы;
- статический метод, оборачивающий конструктор класса.

Итак, выделяют следующие фабрики:

- создающий метод;
- статический фабричный метод;
- простая фабрика;
- фабричный метод;
- абстрактная фабрика.

Рассмотрим каждую из них в подробности.

Создающий метод – это простой метод-обертка над вызовом конструктора продукта. Выделив создающий метод, изолируются любые изменения в конструировании продуктов от основного кода. Например, можно вовсе убрать вызов конструктора из создающего метода, отдавая вместо нового некий существующий объект.

Многие называют его фабричным методом, только потому, что он создает новые объекты. Ошибочное мнение: «этот метод создает объекты, а раз все фабрики создают что-то, значит этот метод – фабричный». И это вносит основную путаницу между понятием Создающего метода и паттерном Фабричный метод.

В этом примере метод next является создающим методом:

Листинг 1 – Пример создающего метода

```
class Number {  
    private $value;  
  
    public function __construct($value) {  
        $this->value = $value;  
    }  
  
    public function next() {  
        return new Number ($this->value + 1);  
    }  
}
```

Статический фабричный метод – вариация создающего метода, объявленная как static. Если этот метод создает объекты своего же класса, то, он выступает в роли альтернативного конструктора.

Это может быть полезно, если:

- требуется создать разные по функциональности конструкторы, у которых бы совпадали сигнатуры, например, Random(int max) и Random(int min). Это невозможно во многих языках программирования, но создав статический метод, можно обойти это ограничение;

- хочется повторно использовать готовые объекты, вместо создания новых, например, паттерн Одиночка. При вызове конструктора всегда создается новый объект. Это можно обойти, если вынести вызов конструктора в новый метод. В этом методе можно сначала поискать готовый объект в кеше, и только если его нет, создать новый объект.

В данном примере метод `load` является статическим фабричным методом – он предоставляет удобный способ загрузить пользователя из базы данных:

Листинг 2 – Пример статического фабричного метода

```
class User {
    private $name, $email, $phone;

    public function __construct($name, $email, $phone) {
        $this->name = $name;
        $this->email = $email;
        $this->phone = $phone;
    }

    public static function load($id) {
        list($name, $email, $phone) = DB::load_data('users', 'name', 'email', 'phone');
        $user = new User($name, $email, $phone);
        return $user;
    }
}
```

Паттерн Простая фабрика – это класс, в котором есть один метод с большим условным оператором, выбирающим создаваемый продукт. Этот метод вызывают с неким параметром, по которому определяется, какой из продуктов нужно создать. У простой фабрики, как правило, нет подклассов.

Обычно, простую фабрику путают с общим понятием Фабрики или с любым из фабричных паттернов. Однако если объявить класс простой фабрики абстрактным, это не делает его одним и тем же, что и абстрактная фабрика.

Пример простой фабрики представлен на листинге 3.

Листинг 3 – Пример простой фабрики

```
class UserFactory {
    public static function create($type) {
        switch ($type) {
            'user': return new User();
            'customer': return new Customer();
            'admin': return new Admin();
            default:
                throw new Exception("Wrong user type passed.");
        }
    }
}
```

Паттерн Фабричный метод – это устройство классов, при котором подклассы могут переопределять тип создаваемого в суперклассе продукта. Если имеется иерархия продуктов и абстрактный создающий метод, который переопределяется в подклассах, то это паттерн Фабричный метод (листинг 4).

Листинг 4 – Пример фабричного метода

```
class Department {
    public abstract function createEmployee($id);

    public function fire($id) {
        $employee = $this->createEmployee($id);
        $employee->paySalary();
        $employee->dismiss();
    }
}

class ITDepartment extends Department {
    public function createEmployee($id) {
        return new Programmer($id);
    }
}
```

```
}  
  
class AccountingDepartment extends Department {  
    public function createEmployee($id) {  
        return new Accountant($id);  
    }  
}
```

Паттерн Абстрактная фабрика – это устройство классов, облегчающее создание семейств продуктов.

Для примера рассмотрим классы Транспорт + Двигатель + Управление. Вариациями этого семейства могут стать:

Автомобиль + ДвигательВнутреннегоСгорания + Руль

Самолет + РеактивныйДвигатель + Штурвал

Если у вас нет семейств продуктов, значит, абстрактная фабрика не может быть применена.

Заключение

Представлен краткий обзор шаблонов проектирования – фабрики. Приведены примеры реализации данных шаблонов, что показывает отличие между ними. Эта статья может служить отправной точкой в изучении паттернов проектирования, а в частности разобраться в схожих по названию, но разных по содержанию паттернах.

ЛИТЕРАТУРА

1. CPP Rreference [Электронный ресурс] Паттерны проектирования (Design Patterns). – Режим доступа: <http://cpp-reference.ru/patterns/>. – Дата доступа: 25.09.2017.
2. Tproger [Электронный ресурс] Шаблоны проектирования. – Режим доступа: <https://tproger.ru/translations/design-patterns-for-beginners/>. – Дата доступа: 25.09.2017.
3. Technerium [Электронный ресурс] Фабрика – паттерн разработки. – Режим доступа: <http://www.technerium.ru/izuchenie-java-na-praktike/fabrika-pattern-razrabotki>. – Дата доступа: 26.09.2017.

УДК 004

НЕЙРОННЫЕ СЕТИ И МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ**М.А. ИВАНОВ***(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)**Рассматривается принцип использования нейронных сетей в контексте мобильных приложений.*

Человечество давно ведет разработки в области искусственного разума. Нейронные сети – одна из таких разработок. Принципы построения таких сетей заключаются в «копировании» человеческой нервной системы. Их главная особенность в том, что они способны к самообучению, то есть могут работать на основании данных, полученных ранее [1].

Данная работа поможет получить представление о целесообразности и примерах использования нейронных сетей в занимающих значимую долю рынка мобильных приложениях.

Нейросети это не только имитация деятельности систем человека. Они по своей структуре напоминают нервную систему – состоят из отдельных вычислительных элементов, которые можно сравнить с нейронами человека. Все данные, которые получает такая сеть, последовательно проходят обработку нескольких сегментов нейросети (слоев), состоящих из нескольких «нейронов».

Так как сети известны параметры каждого «нейрона» и всей сети в целом, она может вследствие полученных ранее данных менять свою работу, то есть «самообучаться».

Сегодня нейросети применяются довольно часто. К примеру, многие крупные интернет-магазины используют их для того, чтобы предлагать своим клиентам более подходящие для них товары. Кроме того, такие математические модели широко применяются для распознавания и воспроизведения речи, а также распознавания и обработки изображений.

Увеличение спроса на беспилотные автомобили, в свою очередь, открыли новую сферу применения нейронных сетей. Они используются для прокладки маршрута и работы систем навигации.

На нейронные сети обратили внимание и производители антивирусного ПО. Такие разработки в области искусственного разума позволяют защитить информацию от киберпреступников и выявить противозаконный контент в Интернете [2].

Уже сейчас футурологи «рисуют» удивительную картину будущего, где масштабы применения нейросетей возрастут многократно. Например, появятся роботы-садоводы, которые благодаря сотням встроенных микрокамер будут получать данные, анализировать их и отделять сорняки от культурных растений. Благодаря «самообучению» такие роботы способны применить к каждому растению индивидуальную обработку.

В разрабатываемом сейчас «интернете вещей» нейронные сети смогут создать голосовые интерфейсы для «общения» различных объектов такой «паутины». Алгоритмы данной математической модели смогут заменить человека в колл-центрах, а также контролеров на складах, переводчиков, консультантов и представителей других профессий. По мнению многих аналитиков, на это нейронным сетям потребуется всего 3–5 лет.

Про искусственный интеллект человечество задумалось очень давно. Об использовании нейронных сетей для решения этого вопроса заговорили 70 лет тому назад. Но, лишь сегодня это стало возможным в тех объемах, которые необходимы.

Популярность нейросетей в настоящее время обусловлена возросшей скоростью работы «нейронов», то есть отдельных вычислительных элементов, вследствие этого удалось снизить время на стадию «обучения». Требуемые для работы таких сетей десятки миллионов данных благодаря современным процессорам и видеокартам обрабатываются достаточно быстро. Современные технологии ускоренного обучения и вывели их на передовую искусственного интеллекта.

В последнее время на рынке мобильных приложений появилось очень много развлекательных проектов, которые используют нейросети. Главную рекламу нейросетям сделало приложение Prisma. Кроме того, стоит отметить и конкурента этого приложения Mlvch, а также популярный видеосервис MSQRD, который также использует описываемые математические алгоритмы.

Приложение Prisma, созданное российскими разработчиками, преобразующее фотографии под картины известных художников, пожалуй, самое известное применение нейронных сетей в современной жизни. В первую очередь, Prisma демонстрирует возможности данных математических алгоритмов при разработке мобильных приложений. Но, такая стилизация изображений может применяться не только в развлекательных целях, но и в дизайне, мультипликации, компьютерной графике и других подобных областях.

При обработке фотографии определяются отображенные на ней объекты. После чего к снимку применяется стиль выбранного художника. То есть, приложение с помощью искусственных нейронных сетей «дублирует» работу мозга художника.

Приложение Mlvch является главным конкурентом Prisma. Оно работает на подобном алгоритме, но отличается более сложной проработкой. Если в Prisma изображение обрабатывается за 20–30 итераций, то в Mlvch за 100, что позволяет на выходе получить более детальное изображение.

Конечно, прикладные продукты, использующие нейросети, чаще всего применяют для анализа изображений. Но, с помощью приложения Clarifai можно пойти дальше и распознать изображение на видео. Еще один подобный сервис предлагает российская компания Ntechlab. Она разработала приложение Findface, с помощью которого можно по фотографии пользователя найти его профиль в социальной сети «В контакте».

Сервис «Авто.ру», который сегодня принадлежит Яндексу, по фотографии самостоятельно распознает марку и модель изображенного автомобиля. Принцип «компьютерного зрения», применяемый в таком распознавании, также работает на технологии нейросетей.

С помощью приложения Ostagram можно не только накладывать стили, но даже перерисовывать картины. А с помощью Deepomatic, пользователю на основе того, какие картинки он раздавал в интернете, предлагаются товары из интернет-магазинов.

Конечно, анализом картинок с рекламными и развлекательными целями область применения нейросетей не ограничивается. Стартап Ava позволяет следить за своим здоровьем. Приложение анализирует фотографию блюд и продуктов питания и выдает информацию о калориях и различных добавках.

Не так давно с помощью алгоритма нейронных сетей Яндекс записал музыкальный альбом «Нейронная оборона». Проект, который получил название «Автопоэт», генерировал зарифмованные тексты.

Использует описываемые математические алгоритмы и другой ведущий интернет-поисковик Google. В ноябре 2015 года эта компания внедрила в свою почту Gmail специальную функцию, благодаря которой сервис сам генерирует три ответа на входящее письмо. Выбрать лучший из них можно одним кликом мыши.

Также нейронные сети используют при создании беспилотных автомобилей и летающих аппаратов. Сначала этими средствами передвижения управляет человек, далее данные собираются в специальную базу, которая используется для «обучения» автопилота. Уже сейчас в Швейцарии действует беспилотный летательный объект, который использует принципы нейросети и применяется для поиска заблудившихся туристов [3].

Заключение

Проанализировав приведенную выше информацию, можно сказать о том, что будущее действительно за технологиями, использующими искусственный интеллект и нейронные сети. Уже на данный момент существует множество приложений (мобильных в том числе), работающих при использовании технологий нейросетей. Области применения данной технологии простираются от фотографии и дизайна до действительно важных, имеющих высокую практическую ценность приложений в сфере медицины, здравоохранения, беспилотных аппаратов и т.д.

ЛИТЕРАТУРА

1. Википедия – свободная общедоступная мультязычная универсальная интернет-энциклопедия [Электронный ресурс] / Искусственная_нейронная_сеть. – Режим доступа: [https://ru.wikipedia.org/wiki/ Искусственная_нейронная_сеть/](https://ru.wikipedia.org/wiki/Искусственная_нейронная_сеть/). – Дата доступа: 24.09.2017.
2. HighLoad++ – Профессиональная конференция разработчиков высоконагруженных систем [Электронный ресурс] / Нейронные сети: практическое применение. – Режим доступа: <http://www.highload.ru/2016/abstracts/2417.html/>. – Дата доступа: 26.09.2017.
3. Хабрахбр [Электронный ресурс] / Нейронные сети: практическое применение. – Режим доступа: <https://habrahabr.ru/post/322392/>. – Дата доступа: 25.09.2017.

УДК 004

ТЕХНОЛОГИИ РАБОТЫ НЕЙРОННЫХ СЕТЕЙ

М.А. ИВАНОВ

(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

Рассматривается принцип построения и работы нейронных сетей. Представлены понятия «нейронная сеть», «нейрон», «синапс».

Многие материалы по нейронным сетям сразу начинаются с демонстрации довольно сложных архитектур. При этом самые базовые вещи, касающиеся функций активаций, инициализации весов, выбора количества слоев в сети и т.д., если и рассматриваются, то вскользь. Получается, начинающему практику нейронных сетей приходится брать типовые конфигурации и работать с ними фактически вслепую.

В данной статье рассмотрим механизм работы нейронных сетей, начиная с основ, с самой простой конфигурации. Это позволит наработать практическую интуицию в построении архитектур нейросетей, которая на практике оказывается очень ценным активом [1].

Нейронная сеть – это последовательность нейронов, соединенных между собой синапсами. Схематическое представление нейронов показано на рисунке 1. Структура нейронной сети пришла в мир программирования прямоком из биологии. Благодаря такой структуре, машина обретает способность анализировать и даже запоминать различную информацию. Нейронные сети также способны не только анализировать входящую информацию, но и воспроизводить ее из своей памяти. Другими словами, нейросеть – это машинная интерпретация мозга человека, в котором находятся миллионы нейронов, передающих информацию в виде электрических импульсов.

Базовый тип нейронных сетей – это сеть прямого распространения (далее СПР). Это сеть с последовательным соединением нейронных слоев, в ней информация всегда идет только в одном направлении.

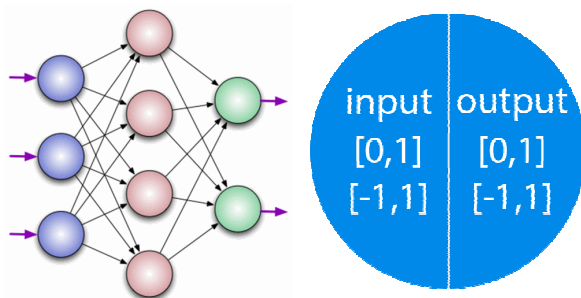


Рисунок 1. – Схематическое представление нейронов

Нейрон – это вычислительная единица, которая получает информацию, производит над ней простые вычисления и передает ее дальше. Они делятся на три основных типа: входной, скрытый и выходной (рис. 2). В том случае, когда нейросеть состоит из большого количества нейронов, вводят термин слоя. Соответственно, есть входной слой, который получает информацию, и скрытых слоев (обычно их не больше 3), которые ее обрабатывают, и выходной слой, который выводит результат. У каждого из нейронов есть 2 основных параметра: входные данные (inputdata) и выходные данные (outputdata). В случае входного нейрона: $\text{input} = \text{output}$. В остальных, в поле input попадает суммарная информация всех нейронов с предыдущего слоя, после чего она нормализуется с помощью функции активации и попадает в поле output.

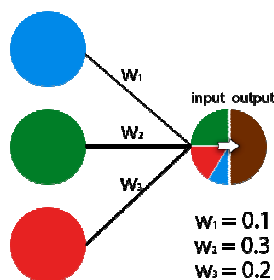
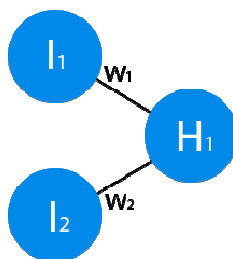


Рисунок 2. – Схематическое представление синапса

Синапс – это связь между двумя нейронами. У синапсов есть 1 параметр – вес. Благодаря чему входная информация изменяется, когда передается от одного нейрона к другому. К примеру, есть 3 нейрона, которые передают информацию следующему. В таком случае, существуют 3 веса, соответствующие каждому из этих нейронов. У того нейрона, у которого вес будет больше, та информация и будет доминирующей в следующем нейроне (пример – смешение цветов). На самом деле, совокупность весов нейронной сети или матрица весов – это своеобразный мозг всей системы. Именно благодаря этим весам, входная информация обрабатывается и превращается в результат [1].

Важно помнить, что во время инициализации нейронной сети веса расставляются в случайном порядке.

В примере на рисунке 3 изображена часть нейронной сети, где буквами I обозначены входные нейроны, буквой H – скрытый нейрон, а буквой w – веса.



$$1) H_{input} = (I_1 * w_1) + (I_2 * w_2)$$

$$2) H_{output} = f_{activation}(H_{input})$$

Рисунок 3. – Пример части нейронной сети

Из формулы видно, что входная информация – это сумма всех входных данных, умноженных на соответствующие им веса. Тогда дадим на вход 1 и 0. Пусть $w_1 = 0,4$ и $w_2 = 0,7$

Входные данные нейрона H_1 будут следующими: $1 \times 0,4 + 0 \times 0,7 = 0,4$. Теперь, когда существуют входные данные, есть возможность получить выходные данные, подставив входное значение в функцию активации. Выходные данные передаются дальше. Так повторяется для всех слоев до выходного нейрона. Запустив такую сеть в первый раз можно увидеть, что ответ далек от правильного, потому что сеть не натренирована. Для улучшения результатов сеть необходимо тренировать.

Функция активации – это способ нормализации входных данных. Если на входе будет большое число, пропустив его через функцию активации, на выходе число попадает в нужный диапазон. Функций активации достаточно много, имеет смысл рассмотреть самые основные: Сигмоид (Логистическая) и Гиперболический тангенс. Главные их отличия – это диапазон значений.

Логистическая функция (рис. 4) – самая распространенная функция активации, ее диапазон значений $[0, 1]$. Именно на ней показано большинство примеров в сети. Соответственно, если в конкретном случае присутствуют отрицательные значения (например, акции могут идти не только вверх, но и вниз), понадобится функция, которая захватывает и отрицательные значения.

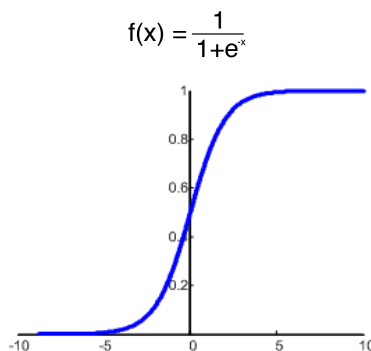


Рисунок 4. – Логистическая функция активации

Имеет смысл использовать гиперболический тангенс (рис. 5), только тогда, когда значения могут быть и отрицательными, и положительными, так как диапазон функции $[-1; 1]$. Использовать эту функ-

цию только с положительными значениями нецелесообразно, так как это значительно ухудшит результаты нейросети [3].

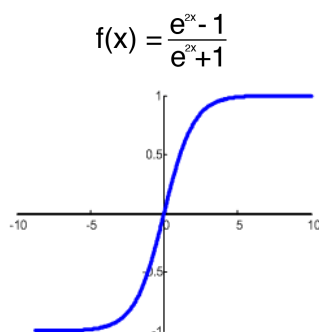


Рисунок 5. – Гиперболический тангенс

Тренировочный сет – это последовательность данных, которыми оперирует нейронная сеть.

Итерация – это своеобразный счетчик, который увеличивается каждый раз, когда нейронная сеть проходит один тренировочный сет. Другими словами, это общее количество тренировочных сетов, пройденных нейронной сетью.

Эпоха – это величина, которая устанавливается в 0 при инициализации нейронной сети и имеет потолок, задаваемый вручную. Чем больше эпоха, тем лучше натренирована сеть и соответственно, ее результат. Эпоха увеличивается каждый раз, когда проходит весь набор тренировочных сетов.

Важно не путать итерацию с эпохой и понимать последовательность их инкремента. Сначала n раз увеличивается итерация, а потом эпоха и никак не наоборот. Нельзя сначала тренировать нейросеть только на одном сете, потом на другом и т.д. Нужно тренировать каждый сет один раз за эпоху. Данным способом можно избежать ошибок в вычислениях.

Ошибка – это процентная величина, отражающая расхождение между ожидаемым и полученным ответами. Ошибка формируется каждую эпоху и должна идти на спад. Если этого не происходит, значит, нейросеть спроектирована не лучшим образом.

Принцип подсчета ошибки во всех случаях одинаков. За каждый сет от идеального ответа отнимается полученный. Далее, либо возводится в квадрат, либо вычисляется квадратный тангенс из этой разности, после чего полученное число делится на количество сетов [2].

Заключение

Нейронные сети – это мощный, но при этом нетривиальный прикладной инструмент. Лучший способ научиться строить рабочие нейросетевые конфигурации – начинать с более простых моделей и экспериментировать, набирая опыт и интуицию практики нейронных сетей.

ЛИТЕРАТУРА

1. Искусственная_нейронная_сеть // Википедия – свободная общедоступная мультязычная универсальная интернет-энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Искусственная_нейронная_сеть/. – Дата доступа: 24.09.2017.
2. Нейронные сети в картинках: от одного нейрона до глубоких архитектур [Электронный ресурс] // Хабрахабр. – Режим доступа: <https://habrahabr.ru/post/322438/>. – Дата доступа: 25.09.2017.
3. Самое главное о нейронных сетях. Лекция в Яндексе [Электронный ресурс] // Хабрахабр. – Режим доступа: <https://habrahabr.ru/company/yandex/blog/307260/>. – Дата доступа: 25.09.2017.

УДК 004.62

РАЗРАБОТКА УРОВНЯ РАБОТЫ С БАЗОЙ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКА МУВАТИС В ВЕБ-ПРИЛОЖЕНИИ «ОНЛАЙН СЕРВИС ДЛЯ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЯ НА JAVA»

П.А. СТАНКЕВИЧ

(Представлено: Ю.Н. КРАВЧЕНКО)

Рассматривается разработка уровня работы с базой данных с использованием фреймворка MyBatis в веб-приложении, написанном на языке программирования Java. Данное веб-приложение реализовано с использованием множества фреймворков, одними из которых стали Spring Framework, Spring Security и MyBatis.

На стадии проектирования объемных веб-приложений часто возникает вопрос, как необходимо работать с уровнем базы данных. Многие останавливаются на стандартных средствах работы с данными, вследствие чего замедляют скорость работы веб-приложения, делают исходные файлы веб-приложения громоздкими и сложно читаемыми с множеством повторений одинаковых фрагментов кода, а также нарушают безопасность работы с данными.

Для решения данных проблем существуют специализированные фреймворки, которые с более высокой скоростью обрабатывают запросы пользователей, и имеют большую степень защищенности от SQL-инъекций.

В данной работе пойдет речь о фреймворке MyBatis, который был использован в веб-приложении «Онлайн сервис для обучения программирования на Java».

При проектировании «Онлайн сервиса для обучения программирования на Java» была взята за основу архитектура, представленная на рисунке 1. Для ее реализации необходимо было выбрать способ работы с базой данных. Выбранный способ должен был обладать такими характеристиками, как:

- работа под Java SE 8;
- взаимодействие с фреймворками Spring Framework и Apache Commons DBCP;
- высокую скорость работы.

Архитектура слоев веб-приложения

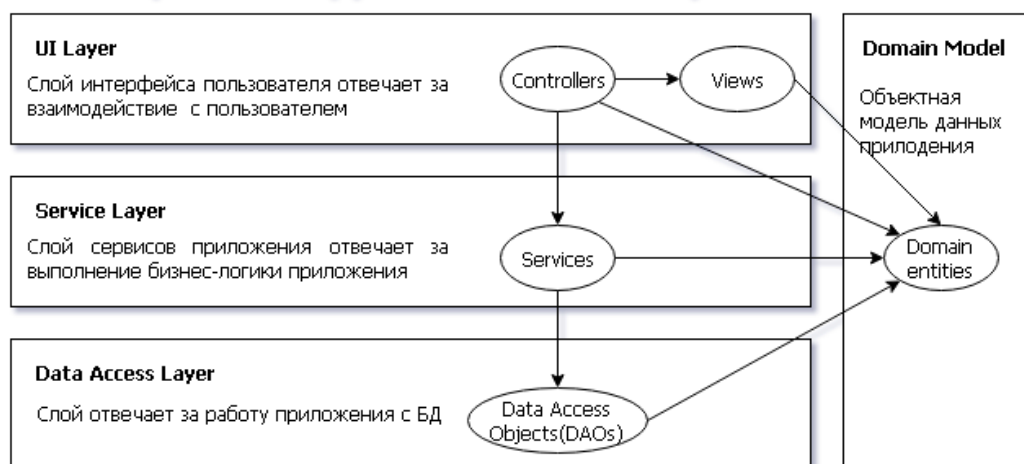


Рисунок 1. – Архитектура разработанного веб-приложения

После рассмотрения возможных вариантов был выбран MyBatis как основное средство на уровне доступа к данным. MyBatis – это фреймворк для связывания данных, который упрощает работу с реляционными базами данных в объектно-ориентированных приложениях [1].

Из истории MyBatis можно подчеркнуть, что изначально он имел название iBATIS, который разрабатывал Клинтон Бегин в 2001 году. Тогда основное внимание уделялось разработке криптографических программных решений. Первым программным продуктом, выпущенным iBATIS, был Secrets, он был предназначен для шифрования личных данных и использовался как инструмент подписи документов. Вскоре после выпуска Secrets проект iBATIS стал фокусироваться на Интернете и других интернет-

технологиях. Благодаря этому были разработаны несколько интересных программных продуктов, в том числе веб-фреймворк Axle, который был альтернативой JSP.

В начале 2002 года Microsoft опубликовала документ, в котором утверждается, что .Net был в 10 раз быстрее и в 4 раза более производительным, чем J2EE. Понимая, что это не так, Клинтон быстро отреагировал, и 1 июля 2002 года вышел аналог программного продукта Pet Store, который получил название JPetStore. Основываясь на тех же требованиях к магазину Pet Store, JPetStore продемонстрировал, что Java может быть не только более продуктивной, чем .NET, но также может достигнуть лучшей архитектуры, чем была использована в реализации Microsoft.

JPetStore использовал интересный уровень сохранения, который быстро привлек внимание сообщества с открытым исходным кодом. Вследствие чего были поражены iBATIS Database Layer и Data Access Object (DAO). Вскоре после выпуска JPetStore вопросы и запросы к базам данных SQL Maps и DAO породили проект, который станет известен как уровень базы данных iBATIS. Уровень данных iBATIS включает в себя две структуры, упакованные вместе: SQL Maps и DAO.

В 2004 году проект присоединился к Apache Software Foundation и дальше разрабатывался в Apache на протяжении 6 лет.

За шесть лет многое изменилось в мире программного обеспечения с открытым исходным кодом. Все от практики развития инфраструктуры, лицензирования до технологий баз. В 2010 году основная команда разработчиков решила, что эти разработки стоит изучить и что iBATIS выиграет от некоторых значительных изменений. Это было серьезное решение, и этот шаг означал, что необходимо оставить имя iBATIS в прошлом [2].

Основным назначением фреймворка считается возможность отвязать Java классы от различных СУБД. Т.е. идея такова – описываем преобразование или маппинг таблиц базы данных или запросов в атрибуты Java классов посредством XML файла и, делая вызовы соответствующих методов MyBatis-a, просто заполняем данными эти Java классы.

Рассмотрим реализацию уровня работы с данными (Data Access Layer) на примере сущности Группа. Реализация сущности (Java-класс) представлена в листинге 1.

Листинг 1 – КлассGroup

```
1 package by.psu.learnjava.entity;
2
3 public class Group extends Entity {
4
5     private static final long serialVersionUID = 1L;
6
7     private User owner;
8     private String accessKey;
9     private String name;
10    private List<User> members;
11
12    public User getOwner() {
13        return owner;
14    }
15
16    public void setOwner(User owner) {
17        this.owner = owner;
18    }
19    ...
20 }
```

Из листинга 1 можно сделать вывод, что класс наследуются от некоторого класса Entity (Сущность). Класс Group имеет поля характерные для своей реализации, а также имеет методы получения и изменений полей, гетеры и сетеры (getters, setters). В листингах вместо троеточий пропущены гетеры и сетеры для остальных полей.

Далее необходимо создать слой Data Access Object (DAO). DAO – это шаблон проектирования, который предоставляет определенные возможности независимо от того, какой механизм хранения используется и без необходимости специальным образом соответствовать этому механизму хранения. В качестве примера рассмотрим интерфейс GroupDao, приведенный в листинге 2.

Листинг 2 – Интерфейс GroupDao

```
21 package by.psu.learnjava.dao;
22
23 public interface GroupDao extends CrudDao<Long, Group> {
24
25     List<Group>findAll();
26
27     List<Group>findForUserId(Long id);
28
29     List<User>findMembersByGroupId(Long id);
30
31     Group findByToken(String token);
32
33     void deleteMember(Map<String, Object> map);
34
35     void updateGroupList(Map<String, Object> map);
36 }
```

Из листинга 2 можно заметить, что интерфейс расширяется с помощью CrudDao, и имеет дополнительно 6 операций, поиска всех групп, поиска групп по идентификатору пользователя, поиск пользователей по идентификатору группы, поиск группы по токен-ключу, удаление пользователя из группы, а также обновление списка пользователей в группе.

Каждый уважающий себя программист знает, нет смысла интерфейса без его реализации. Нужно создать класс, который наследуется от SqlSessionDaoSupport и реализует интерфейс GroupDao. SqlSessionDaoSupport – это класс MyBatis'a в нем есть объект DataSource, в котором хранятся данные подключений. Для реализации DataSource в данном примере используется библиотека ApacheCommonsDBCP, а именно класс BasicDataSource. Реализация класса представлена в листинге 3.

Листинг 3 – Класс GroupMapper

```
1 package by.psu.learnjava.dao.mybatis;
2
3 public class GroupMapper extends SqlSessionDaoSupport implements GroupDao {
4
5     @Override
6     public void create(Group obj) {
7         getSqlSession().insert("Group.create", obj);
8     }
9
10    @Override
11    public Group read(Long id) {
12        return getSqlSession().selectOne("Group.read", id);
13    }
14
15    @Override
16    public void update(Group obj) {
17        getSqlSession().update("Group.update", obj);
18    }
19
20    @Override
21    public void delete(Long id) {
22        getSqlSession().delete("Group.delete", id);
23    }
24
25    @Override
26    public List<Group>findAll() {
27        return getSqlSession().selectList("Group.findAll");
28    }
29 }
```

```
30  @Override
31  public List<Group>findForUserId(Long id) {
32      return getSqlSession().selectList("Group.findForUserId", id);
33  }
34
35  @Override
36  public List<User>findMembersByGroupId(Long id) {
37      return getSqlSession().selectList("Group.findMembersByGroupId", id);
38  }
39
40  @Override
41  public Group findByToken(String token) {
42      return getSqlSession().selectOne("Group.findByToken", token);
43  }
44
45  @Override
46  public void deleteMember(Map<String, Object> map) {
47      getSqlSession().delete("Group.deleteMember", map);
48  }
49
50  @Override
51  public void updateGroupList(Map<String, Object> map) {
52      getSqlSession().update("Group.updateGroupList", map);
53  }
54 }
```

Из приведенного примера в листинге 3 можно заметить, что в каждом методе вызывается `getSqlSession`, после чего могут быть вызваны различные методы, благодаря которым будут изменены или получены данные. Но в параметрах у всех методов указана ссылка, которая указывает, в каком `xml`-маппере лежит нужный запрос к БД.

Для работы с `MyBatis` необходимо создать файл конфигурации, его часть приведена в листинге 4. Он описывается с помощью `xml` файла, в котором содержится основной тег `configuration`, для его написания необходимо описать тег `typeAliases` и `mappers`. Тег `typeAliases` должен содержать в себе теги `typeAlias`, который содержит ссылку на тип, и альтернативную ссылку на этот тип. Тег `mappers` должен содержать пути к файлам с описаниям действий над сущностью.

Листинг 4 – Конфигурация `MyBatis`

```
1  <?xml version = "1.0" ?>
2  <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
3  "http://mybatis.org/dtd/mybatis-3-config.dtd">
4
5  <configuration>
6  <typeAliases>
7      ...
8  <typeAliastype="by.psu.learnjava.entity.Group" alias="group" />
9      ...
10 </typeAliases>
11 <mappers>
12     ...
13 <mapper resource="by/psu/learnjava/dao/mybatis/Group.xml" />
14     ...
15 </mappers>
16 </configuration>
```

После настройки файла конфигурации необходимо реализовать файл с описанием возможных действий над сущностью. Данный файл описывается в формате `xml`, в котором корневым тегом служит

mapper с указанием имени маппера. Внутри доступны resultMap, sql, select, delete, insert, update. Тер resultMap описывает сущность, sql позволяет хранить фрагменты sqlкода, select нужен для считывания данных, а delete, insert, update необходимы для удаления, добавления и обновления записей соответственно. Пример файла приведен в листинге 5.

Листинг 5 – Фрагмент файла Group.xml

```
1  <?xml version = "1.0" ?>
2  <!DOCTYPE mapper
3    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5  <mapper namespace="Group">
6
7    <resultMap type="group" id="group">
8      <id property="id" column="group_id" />
9      <result property="accessKey" column="group_access_key" />
10     <result property="name" column="group_name" />
11     <collection property="owner" javaType="User" resultMap="User.user"/>
12   </resultMap>
13
14     <sql id="sql">
15       SELECT
16         g.id AS group_id,
17         g.access_key AS group_access_key,
18         g.name AS group_name,
19         u.id AS user_id,
20         u.email AS user_email,
21         u.passwd AS user_passwd,
22         u.login AS user_login,
23         u.unconfirmed_email AS user_unconfirmed_email,
24         r.id AS role_id,
25         r.name AS role_name,
26         ls.id AS learn_speed_id,
27         ls.speed AS learn_speed_speed,
28         ls.name AS learn_speed_name,
29         p.id AS person_id,
30         p.name AS person_name,
31         p.surname AS person_surname,
32         p.patronymic AS person_patronymic,
33         p.birthday AS person_birthday
34
35     FROM
36     learn_java.groups AS g
37       INNER JOIN learn_java.users AS u ON g.id_user=u.id
38       INNER JOIN learn_java.roles AS r ON u.id_role=r.id
39       INNER JOIN learn_java.learn_speed AS ls ON u.id_learn_speed=ls.id
40       INNER JOIN learn_java.persons AS p ON u.id_person=p.id
41     </sql>
42
43   <insert id="create" parameterType="group">
44     <selectKey keyProperty="id" order="BEFORE" resultType="long">
45       SELECT nextval('learn_java.groups_id_seq'::regclass)
46     </selectKey>
47     INSERT INTO learn_java.groups(
48       id, id_user, name)
49     VALUES ({id}, #{user.id}, #{name})
50   </insert>
```

```
51
52 <select id="read" parameterType="long" resultMap="group">
53     <include refid="sql" />
54     g.id = #{id}
55 </select>
56
57 <select id="findAll" resultMap="group">
58 <include refid="sql" />
59     ORDER BY g.id
60 </select>
61
62 <select id="findForUserId" parameterType="long" resultMap="group">
63 <include refid="sql" />
64     WHERE u.id = #{id}
65     ORDER BY g.id
66 </select>
67 ...
68 </mapper>
```

Заключение

Разработка веб-приложений с использованием паттерна DAO, а также фреймворка MyBatis актуальна, так как данный способ работы с данными давно доказал свою эффективность, а также не зависимость от выбранной СУБД. Но данный тип реализации уровня работы с данными лучше использовать с объемными проектами, которые, скорее всего, нужно будет поддерживать на протяжении некоторого времени.

ЛИТЕРАТУРА

1. ApacheAttic [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Apache_Attic. – Дата доступа: 20.09.2017.
2. TheMyBatisBlog [Электронный ресурс]. – Режим доступа: <http://blog.mybatis.org/p/about.html>. – Дата доступа: 20.09.2017.

УДК 004.62

ВНЕДРЕНИЕ СИСТЕМЫ АВТОРИЗАЦИИ И АУТЕНТИФИКАЦИИ ПОЛЬЗОВАТЕЛЕЙ НА ОСНОВЕ ФРЕЙМВОРКА SPRINGSECURITY В ВЕБ-ПРИЛОЖЕНИИ «ОНЛАЙН СЕРВИСА ДЛЯ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЯ НА JAVA»

П.А. СТАНКЕВИЧ

(Представлено: Ю.Н. КРАВЧЕНКО)

Рассматривается настройка фреймворка SpringSecurity в качестве механизма обеспечения аутентификации и авторизации пользователей для разработанного онлайн сервиса. Данное веб-приложение было реализовано с использованием множества фреймворков, одними из которых стали Spring Framework, Spring Security и MyBatis.

При создании веб-приложений, в которых существуют различные роли пользователей, часто возникает вопрос, каким способом создавать авторизацию, где ее хранить, как аутентифицировать и авторизовывать пользователей. Для этого программисты практически всегда используют различные фреймворки. Об одном из таких фреймворков, а именно SpringSecurity, который был внедрен в онлайн сервис для обучения программированию на Java и пойдет речь.

Основная часть

На стадии проектирования веб-приложения были составлены основные требования для обеспечения аутентификации и авторизации пользователей, а именно:

- взаимодействие с фреймворками SpringFramework и ApacheCommonsDBCP;
- высокая скорость работы;
- возможность описания файла конфигурации в XML формате.

В процессе анализа возможных вариантов был выбран фреймворк SpringSecurity, так как он полностью соответствует перечисленным требованиям и имеет достаточно детализированные возможности по настройке.

SpringSecurity это Java/JaVaEE фреймворк, предоставляющий механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности для промышленных приложений, созданных с помощью SpringFramework. Проект был начат Беном Алексом (BenAlex) в конце 2003 года под именем «AcegiSecurity» и был публично представлен под лицензией ApacheLicense в марте 2004. Впоследствии был включен в Spring как официальный дочерний проект. Он был впервые публично представлен под новым именем SpringSecurity 2.0.0 в апреле 2008 года, что включило официальную поддержку и подготовку от SpringSource [1].

«Онлайн сервис для обучения программированию на Java» должен был обладать 3 различными ролями пользователей: модератор, администратор и пользователь.

Пользователь должен был иметь следующие права:

- просмотр и редактирование личной информации;
- просмотр обучающих материалов;
- просмотр магазина и осуществление покупок за внутри-сервисную валюту;
- прохождение тестов;
- отправление жалоб на вопросы из-за найденных ошибок;
- создание и участие в группах.

Каждому создателю группы было доступно меню просмотра и управления группой.

Модератору должны быть присвоены такие права, как:

- просмотр и редактирование обучающих материалов;
- редактирование цен на внутрисервисную валюту;
- просмотра диаграмм, отражающих актуальность товаров в магазине;
- редактирование количества начисляемой внутри-сервисной валюты за различные действия пользователя;
- просмотр жалоб пользователей;
- подтверждение или отклонение найденной пользователем ошибки.

Администратор должен обладать всеми возможностями модератора, а также возможностью управлять модераторами.

Рассмотрим реализацию конфигурации SpringSecurity в разработанном веб-приложении.

Часть файла конфигурации представлена в листинге 1.

Листинг 1 – Файл конфигурации SpringSecurity

```

37 <?xml version="1.0" encoding="UTF-8"?>
38 <b:beansxmlns="http://www.springframework.org/schema/security"
39 xmlns:b="http://www.springframework.org/schema/beans"
40 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
41 xsi:schemaLocation="
42 http://www.springframework.org/schema/security
43 http://www.springframework.org/schema/security/spring-security.xsd
44 http://www.springframework.org/schema/beans
45 http://www.springframework.org/schema/beans/spring-beans.xsd">
46
47 <http realm="Learn Java" use-expressions="false">
48 <intercept-url pattern="/index.html*" access="IS_AUTHENTICATED_ANONYMOUSLY"
49 />
50 <intercept-url pattern="/login.html*" access="ROLE_ANONYMOUS" />
51 <intercept-url pattern="/logout.html*" access="IS_AUTHENTICATED_ANONYMOUSLY"
52 />
53 <intercept-url pattern="/register.html*" access="ROLE_ANONYMOUS" />
54 <intercept-url pattern="/css/**" access="IS_AUTHENTICATED_ANONYMOUSLY" />
55 <intercept-url pattern="/images/**" access="IS_AUTHENTICATED_ANONYMOUSLY" />
56 <intercept-url pattern="/fonts/**" access="IS_AUTHENTICATED_ANONYMOUSLY" />
57 <intercept-url pattern="/js/**" access="IS_AUTHENTICATED_ANONYMOUSLY" />
58 <intercept-url pattern="/profile.html*" access="ROLE_USER, ROLE_MODER,
59 ROLE_ADMIN" />
...
30 <intercept-url pattern="/addModer.html*" access="ROLE_ADMIN" />
...
38 <intercept-url pattern="/**" access="ROLE_USER" />
39 <form-login login-page="/login.html" username-parameter="email" password-
40 parameter="passwd" authentication-failure-url="/login.html?login_error=auth" />
41 <logout logout-success-url="/logout.html" />
42 <csrf />
43 </http>
44
45 <authentication-manager>
46 <authentication-provider>
47 <password-encoder hash="md5"/>
48 <jdbc-user-service data-source-ref="dataSource"
49 users-by-username-query="
50 SELECT LOWER(username) as username, password, enabled
51 FROM learn_java.login(LOWER(?))"
52 authorities-by-username-query="
53 SELECT LOWER(u.email) AS username, r.name AS role
54 FROM learn_java.users u
55 INNER JOIN learn_java.roles r ON u.id_role=r.id
56 WHERE u.email=LOWER(?)" />
57 </authentication-provider>
58 </authentication-manager>
59
60 </b:beans>

```

В первой строке файла расположена XML-декларация. XML-декларация не является обязательной. Однако если она существует, то должна располагаться в первой строке документа, и до нее не должно быть больше ничего, в том числе пробелов.

Со второй по девятую строку листинга идет объявление схемы SpringSecurityConfig для XML документа.

В 11 строке начинается описание тега `http`. В разрабатываемом приложении было использовано всего два параметра, которые являются `realm` и `use-expressions`. Атрибут `realm` указывает, за какую область будет отвечать последующая настройка. `Use-expressions` включает EL-выражения, по умолчанию они включены.

С 12 по 38 строку листинга показаны примеры для указания ролей к определенным ссылкам. Роли `ROLE_USER`, `ROLE_MODER`, `ROLE_ADMIN` являются внедренными разработчиком, а такие роли как `IS_AUTHENTICATED_ANONYMOUSLY`, `ROLE_ANONYMOUS` являются статическими, они реализованы в самом фреймворке. При просмотре документации к `SpringSecurity` можно прочитать, что `IS_AUTHENTICATED_ANONYMOUSLY` хранит ссылку, указывающую на `ROLE_ANONYMOUS` [2].

Тег `form-login` из 39 строки, используется для указания страницы (атрибут `login-page`), на которой пользователю будет предоставлена возможность для авторизации, в атрибутах `username-parameter` и `password-parameter` указываются значения полей ввода логина и пароля, а в атрибуте `authentication-failure-url` можно указать ссылку, куда будет перенаправлен пользователь, если он введет некорректные данные.

В 41 строке указан тег `logout`, в нем есть возможность указать, каким способом будет происходить выход из системы. Все настройки были оставлены по умолчанию, кроме `logout-success-url` – это url, на который будет перенаправлен пользователь для выхода из системы. Тег `csrf` указывает, что необходимо добавить защиту приложения с помощью `CrossSiteRequestForger`.

Для аутентификации необходимо настроить тег `authentication-manager`, которому необходимо задать настройки для `authentication-provider`. `Authentication-provider` загружает пользовательскую информацию из `jdbc-user-service` и сравнивает комбинацию имени пользователя и пароля со значениями, указанными при входе в систему, причем можно указать в атрибуте `password-encoder`, с помощью какого метода хешируется пароль пользователя.

Заключение

Разработка веб-приложений с использованием фреймворка `SpringSecurity` для осуществления аутентификации и авторизации пользователей в онлайн сервисе является подходящим средством, так как предоставляет надежный, скрытый механизм предоставления данных пользователям, имеет достаточно большое количество всевозможных настроек, большинство из которых не были переопределены, так как настройки имеют по умолчанию жесткий стандарт, подходящий для большинства приложений, а также `SpringSecurity` полностью взаимодействует с `SpringFramework` что обеспечило еще большую легкость в его подключении.

ЛИТЕРАТУРА

1. `SpringSecurity` – Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Spring_Security. – Дата доступа: 21.09.2017.
2. `SpringSecurityDocs` [Электронный ресурс]. – Режим доступа: <https://docs.spring.io/spring-security/site/docs/4.2.3.RELEASE/reference/htmlsingle/>. – Дата доступа: 21.09.2017.

УДК 004.05

**ЗАЩИТА ИНФОРМАЦИИ, ПЕРЕДАВАЕМОЙ ПО ПРОТОКОЛУ SOAP.
БЕЗОПАСНОСТЬ WEB-СЕРВИСОВ (WSSECURITY)****Ю.С. КУВЕЦКИЙ***(Представлено: канд. техн. наук И.Б. БУРАЧЕНОК)*

Рассмотрены особенности протокола SOAP, проблемы с которыми можно столкнуться при его использовании и методы их решения. Проведен анализ основных современных способов обеспечения безопасности процесса общения с web-сервисами по сети Internet, их реализации в стандарте WSSecurity. Сделаны выводы о необходимых мерах для обеспечения полной безопасности при использовании web-сервисов.

Во все времена необходимым условием развития человечества являлся процесс автоматизации различного вида его деятельности. Сегодня развитие передовых информационных технологий (ИТ) открывает новые возможности для создания современных автоматизированных систем (АС) во всех сферах народного хозяйства. Безусловно, возникают и проблемы, связанные с необходимостью повышения уровня безопасности таких систем. Причем выбор методов и средств защиты определяется не только важностью обрабатываемой информации, но и составом АС, ее структурой, способами обработки информации, а также количественным и качественным составом пользователей и обслуживающего персонала. Рассмотрим систему, основными задачами которой являются удаленная компиляция и выполнение программного кода (ПК) на различных web-сервисах. Среди разработчиков современных программных приложений наиболее популярна сервисная архитектура – такие приложения гораздо проще поддерживать, легче решается задача масштабирования, но появляется проблема безопасности, так как происходит передача данных по сети. В данной статье мы рассмотрим вариант построения системы, основываясь на протоколе передачи данных SOAP и стандарте обеспечения безопасности WSSecurity.

Во-первых, рассмотрим, что такое SOAP. Протокол SOAP (SimpleObjectAccessProtocol) используется для обмена произвольными структурированными сообщениями в формате XML по распределенной вычислительной среде. Он может использоваться с любым протоколом прикладного уровня, однако чаще всего SOAP используется поверх HTTP (HyperTextTransferProtocol). Его сообщение в формате XML (eXtensibleMarkupLanguage) представлено на рисунке 1 и представляет из себя контейнер (envelope) содержащий заголовок (header) и тело (body), тело в свою очередь может содержать элемент с ошибками и статусом запроса (fault). Microsoft позволяет использовать протокол SOAP при помощи WCF (WindowsCommunicationFoundation) [1].

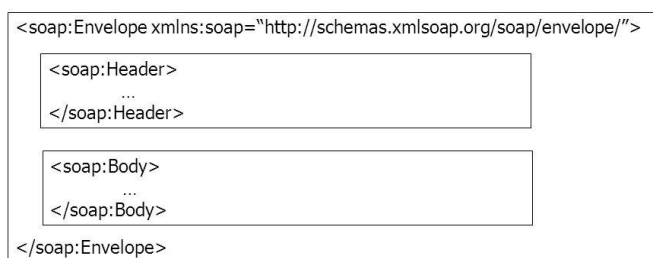


Рисунок 1. – Структура XMLсообщения, передаваемого по протоколу SOAP

Далее рассмотрим три стандарта безопасности применимых к XML: аутентификация, целостность данных, конфиденциальность данных.

Аутентификация гарантирует, что отправитель и получатель являются теми, кем они себя объявляют, доказывает подлинность сторон. Это может реализовываться различными способами. Простой вариант – предоставить идентификатор пользователя и его пароль. Более сложный – это использование сертификата, который содержит все необходимые идентификационные учетные данные и ассоциированную с ними пару из закрытого и открытого ключей.

Для обеспечения целостности информации, которой обмениваются стороны и гарантии того, что содержимое сообщений не будет изменено или повреждено во время их передачи по сети, данные подписывают цифровой подписью (ЦП) с использованием ключей безопасности. Часто для подписания ЦП SOAP-тела запроса используется закрытый ключ сертификата X.509 отправителя. Также можно подписывать SOAP-заголовки запросов, чтобы гарантировать целостность дополнительно передаваемой в транзакции информации.

Третьим требованием для обеспечения безопасности системы является конфиденциальность. Чтобы сделать обмен информацией в запросах и ответах web-сервисов нечитаемым для посторонних, используется технология шифрования. Цель – гарантировать, что любая попытка обратиться к данным при передаче, в памяти или после сохранения потребует соответствующих алгоритмов и ключей безопасности для дешифрования данных.

На данный момент все эти меры безопасности достаточно просто реализуются, а варианты реализации могут зависеть от способа транспортировки сообщений или быть специфичными для протокола SOAP. Каждый из них в отдельности может обеспечить достаточный уровень безопасности, но в таких случаях система всегда подвергается риску. Без аутентификации все будут иметь доступ к сервису. Без ЦП существует вероятность подмены или повреждения информации. При игнорировании правила конфиденциальности, любой злоумышленник сможет проследить всю информацию, проходящую по сети. Если учесть все перечисленные требования, то любую систему можно считать защищенной, при этом не так важны способы их реализации. Конечно, существуют системы, в которых можно пренебречь этими правилами, например, если никак не нужно ограничивать доступ к сервису, то можно опустить аутентификацию. Если масштаб системы ограничивается отправкой запросов между пользователем и одной точкой (сервисом), то достаточным будет использование HTTPS (HyperTextTransferProtocolSecure) и необходимость реализации шифрования отпадает. То же самое и с ЦП данных, в ней нет необходимости, если пересылаемые сообщения не несут какой-либо ценности и не могут навредить системе.

Обеспечение целостности, конфиденциальности и аутентичности сообщения и его отправителя при одновременном сохранении открытости для расширений – основные задачи стандарта WSSecurity [2]. Он не определяет никаких новых технологий, а опирается на уже существующие стандарты, к примеру, XMLEncryption, XMLSignature, сертификаты X.509 или различные криптографические алгоритмы. Благодаря тому, что его базовая концепция основывается на механизмах сообщений, становится возможным обеспечить безопасность из конца в конец (End-to-EndSecurity), например, посредством протокола SSL (SecureSocketLayer), вместо защиты ориентированной на транспорт. Основными элементами стандарта являются следующие базовые механизмы: токены безопасности (ТБ), шифрование, ЦП и отметки о времени.

Токены безопасности применяются при аутентификации, их задача нести подтверждения идентификации (credentials), без которых сама аутентификация невозможна. Чаще всего в роли credentials выступают идентификационный номер (UserID) и соответствующий пароль.

На тему шифрования можно привести много информации, но принципиально различают два механизма шифрования: симметричное и асимметричное. В первом случае для шифрования и дешифровки используется общий ключ, всегда доступный обеим сторонам, этот способ быстрее. Во втором для шифрования и дешифровки применяются разные ключи: личный ключ остается у владельца, а общий ключ распространяется свободно, этот способ более надежный. Оба подхода часто объединяют следующим образом: клиент генерирует симметричный ключ, шифрует им данные любого размера, после чего сам ключ шифруется при помощи асимметричного алгоритма и вкладывается в сообщение.

Цифровые подписи применяются для подтверждения целостности сообщений, благодаря им возможно распознать неправомерные модификации, такие как добавление, изменение или удаление данных.

В WS Security этот подход опирается на стандарт XML DigitalSignature. Принцип ЦП основан на создании контрольных сумм с помощью специальных алгоритмов (дайджест), после чего результаты присоединяются к сообщению и передаются в частично зашифрованном виде.

Стоит также рассмотреть технологию отметок о времени. Дело в том, что в рамках SOA (SimpleObjectAccess) сервисы должны производить определенное действие и таким образом поддерживать взаимодействие без учета состояния (Stateless). Данная особенность позволяет злоумышленникам производить атаки сброса (Replay), когда атакующий повторно отправляет либо сообщения целиком, либо отдельные их части. Для защиты от таких атак необходимо, чтобы каждое сообщение имело свой уникальный идентификатор (MessageID), которые сервис хранит и учитывает при последующих сообщениях.

В результате проведенного исследования можно сделать **вывод**: протокол SOAP достаточно гибок и удобен в использовании; он позволяет абстрагировать общение с различными web-сервисами за счет использования в них общего интерфейса. Благодаря стандарту WSSecurity можно обеспечить достаточный уровень защиты системы от угроз. Для организации общения клиента напрямую с web-сервисами было принято решение ограничиться использованием протокола SSL с достаточно высоким уровнем безопасности для передачи данных по сети напрямую от клиента к сервису.

ЛИТЕРАТУРА

1. Microsoft. Общие сведения о безопасности [Электронный ресурс] / MDN. – Режим доступа: <https://msdn.microsoft.com/>. – Дата доступа: 10.09.2017.
2. IBM Knowledge Center. WS-Security [Электронный ресурс] / IBM. – Режим доступа: <https://www.ibm.com/support/knowledgecenter/>. – Дата доступа: 10.09.2017.

УДК 004.04

ОСОБЕННОСТИ РЕАЛИЗАЦИИ WEB-СЕРВИСОВ,
ОТВЕЧАЮЩИХ ЗА УДАЛЕННУЮ КОМПИЛЯЦИЮ И ВЫПОЛНЕНИЕ ПРОГРАММНОГО КОДА

Ю.С. КУВЕЦКИЙ

(Представлено: канд. техн. наук И.Б. БУРАЧЕНОК)

Разработан универсальный синтаксис шаблонных конструкций для составления различных шаблонов программ. Представлена структура алгоритма для преобразования полученного шаблона в код на современных высокоуровневых языках программирования: C#, Java и Ruby. Рассмотрены особенности реализации web-сервисов, позволяющих компилировать и выполнять переданный им по сети Internet программный код.

На данный момент разработчикам программного обеспечения для того, чтобы составить и запустить простую программу на одном из популярных, высокоуровневых языков программирования, необходимо потратить много времени и сил, особенно если сталкиваешься с этим впервые. Поэтому и возникает необходимость в создании ПО, позволяющего просто и быстро сформировать код, скомпилировать и выполнив его удаленно. *Отличительная особенность* данного ПО – это возможность составить программный код на одном из высокоуровневых языков программирования без знания их синтаксиса. С целью создания легко расширяемого и достаточно универсального ПО, позволяющего работать с различными языками программирования через общие интерфейсы, необходимо решить следующие задачи. Разработать универсальный синтаксис шаблонных конструкций, который позволил бы составлять шаблоны различных программ, а также разработать алгоритм для преобразования полученного шаблона в код на популярных сегодня языках программирования, например, C#, Java и Ruby.

Вначале подробно разберем *синтаксис шаблонных конструкций*. Сами конструкции позволяют выводить данные, писать часть кода программы вручную, а также позволяют описать цикл и условный блок. Перечисленных возможностей достаточно для составления большинства алгоритмов. Детальный анализ уже существующих синтаксисов для построения шаблонов, например, используемые в таких популярных фреймворках, как DjangoTemplates (Python) [1] и Razor (C#) [2], позволил выделить следующие базовые конструкции разрабатываемого синтаксиса:

- произвольный текст, не заключенный в «кодовые скобки» («{%» и «%}»), выводится без изменений;
- конструкция вида «{% программный код %}» приводит к выполнению соответствующего кода;
- конструкция, заключенная в скобки «{%@ выражение %}» и «{%@%}», выражение должно иметь числовой результат, выполняется количество раз, заданное значением выражения;
- конструкция вида «{%= выражение %}» выводит результат выражения любого типа;
- конструкция, заключенная в скобки «{%? выражение %}» и «{%?%}», выражение должно иметь результат типа boolean, выполняется при условии того, что выражение приняло значение истины – true.

На рисунке 1 приведен пример использования шаблонных конструкций с использованием языка программирования C#.

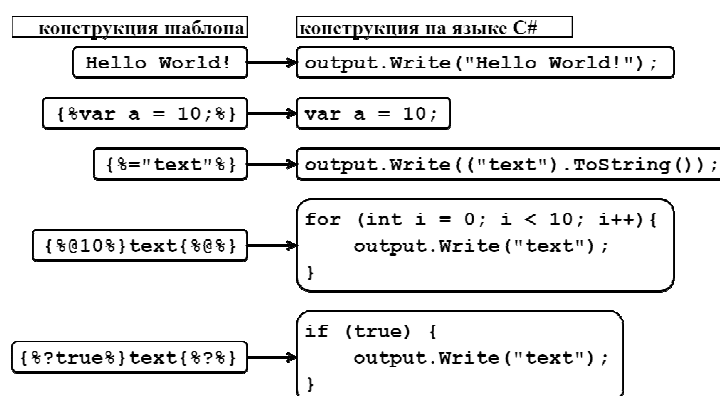


Рисунок 1. – Пример работы шаблонных конструкций

Имея синтаксис шаблона, разработан алгоритм его «парсинга» в программный код. Этот алгоритм основан на использовании регулярных выражений для разбора шаблона по отдельным частям. Плюсы данного способа в том, что из элементарных частей шаблона просто составить программные коды на

различных языках программирования, так как структура программ на выходе алгоритма одна, и их отличия только в синтаксисе и некоторых языковых особенностях.

Изучив все возможности регулярных выражений [3], составление необходимого выражения не представляет сложности. Алгоритм имеет следующую структуру:

1. Шаблон разбивается на части двух типов:
 - части, заключенные в функциональные скобки («%» и «%»);
 - части, вне функциональных скобок.
2. Для каждой из частей определяется ее тип. Это могут быть следующие типы:
 - EndChunk – закрывающие конструкции («%»);
 - TextChunk – части шаблона, которые вне скобок и должны выводиться как есть;
 - ChunkWithBraces – основные элементы шаблона (блок для цикла, условный блок, блок вывода, блок кода программы, блок вывода содержимого).

На рисунке 2 приведено регулярное выражение, отвечающее за первый пункт алгоритма.

```
(?<Braces> \{(?:[\\s\\S]*?)%\})|(?<Text> (?:(?<= % })|(?:^))(?:[\\s\\S]+?)(?: (?= \{)|(?:$)))
```

Рисунок 2. – Регулярное выражение, используемое для разбора шаблона по частям

После того как описанный алгоритм сформировал набор блоков (chunks) из текста шаблона, свою работу начинают алгоритмы, цель которых составить код программы на одном из языков программирования. Данные алгоритмы на первый взгляд очень похожи, однако отличительные особенности, свойственные отдельному языку программирования, все же имеются.

После составления кода программы необходимо выполнить компиляцию кода и его выполнение. Исходя из того, что данный процесс должен происходить удаленно, рассмотрен протокол SOAP как способ реализации удобного и безопасного общения клиента с web-сервисами по сети Internet. Составленный по шаблону код программы отправляется на web-сервис, где компилируется и выполняется. Так как сами языки программирования разнообразны, то реализации нужных сервисов тоже имеют некоторые отличия.

Сервис для компиляции и выполнения программ на языке программирования Java отличается тем, что для каждой новой полученной программы необходимо создавать новый файл на сервере с уникальным именем. Это связано с тем, что компиляция и выполнение программ на языке Java защищено от постоянного нагрузочного компилирования и выполнения при помощи техники кэширования. Если попытаться изменить содержимое ранее созданного на сервере файла и после этого перекомпилировать его и запустить, то выполнится программа, скомпилированная ранее. Однако при создании новых файлов возникает проблема, связанная с необходимостью удаления ненужных. Ее решением является очистка файлов пользователя при завершении его работы.

Веб-сервис для запуска скриптов на языке программирования Ruby работает немного иначе. Тот факт, что язык Ruby – скриптовый язык (т.е. выполняется построчно) убирает необходимость в компиляции, так как этот термин не используется в отношении к скриптам языка Ruby, также отпадает необходимость в создании каких-либо файлов, так как код будет выполняться в eval-пространстве.

В результате проведенного анализа определен синтаксис для составления универсальных шаблонов, использование которых позволит описывать алгоритмы различной сложности, с последующим представлением этих алгоритмов в виде программного кода на различных языках программирования. Также разработан легко расширяемый, независимый от используемого языка программирования алгоритм для разбора шаблона и преобразования в программный код, который обеспечивает высокую производительность и снижает время формирования кода. Кроме этого рассмотрены некоторые особенности реализации web-сервисов, отвечающих за удаленную компиляцию и выполнение программного кода. Указанные особенности, анализ, подходы позволяют осуществить разработку программного обеспечения, отвечающего современным требованиям, что важно при поддержке нескольких высокоуровневых языков программирования и использовании web-сервисов для передачи программного кода и результатов компиляции или выполнения его по сети Internet.

ЛИТЕРАТУРА

1. Django. Templates [Электронный ресурс] / DjangoSoftwareFoundation. – Режим доступа: <https://docs.djangoproject.com/en/1.11/ref/templates/>. – Дата доступа: 15.09.2017.
2. W3Schools.com. WS-Security.ASP.NET Razor – C# and VB Code Syntax [Электронный ресурс] / W3Schools. – Режим доступа: https://www.w3schools.com/asp/razor_syntax.asp. – Дата доступа: 15.09.2017.
3. Lattyf. Тонкости регулярных выражений [Электронный ресурс] // Хабрахабр. – Режим доступа: <https://habrahabr.ru/post/112016/>. – Дата доступа: 16.09.2017.

УДК 004.42

**ЦЕЛЕСООБРАЗНОСТЬ ИСПОЛЬЗОВАНИЯ ИГРОВЫХ СЕРВИСОВ,
ПРЕДОСТАВЛЯЕМЫХ APPLE ДЛЯ РАЗРАБОТКИ ПРИЛОЖЕНИЙ****А.И. ВАЛЬКОВИЧ***(Представлено: канд. техн. наук И.Б. БУРАЧЕНОК)*

Осуществлен сравнительный анализ сервисов, предоставляемых компанией Apple для разработки приложений для платформы iOS и собственных решений. Также автор делает вывод о целесообразности и выгоде от использования внутренних сервисов платформы в процессе разработки небольших приложений.

Современное приложение состоит из большого количества подсистем. Разработчику требуется сохранять данные в облаке, объединять пользователей в общие комнаты, присылать уведомления и выполнять множество других действий, которые являются второстепенными для приложения, но без них продукт окажется скучными не будет выделяться функционалом среди конкурентов. На этапе проектирования возникает вполне очевидный и важный вопрос – реализовывать весь функционал самому или использовать готовое решение? Безусловно, однозначного ответа на этот вопрос нет и не будет, все зависит от ситуации. Однако нужно понимать, что практически для любой задачи можно найти подходящее готовое решение, которое пусть и не полностью решит проблему, но избавит разработчика от необходимости разбираться во всех тонкостях и позволить сэкономить самый главный и ценный ресурс – время.

Компания Apple известна, в первую очередь, как компания, которая продает доступ к своей экосистеме. Большую часть стоимости ее устройств составляют сервисы и программное обеспечение, доступ к которым предоставляется как пользователям, так и разработчикам контента. В первую очередь это касается сервисов и функционала, который доступен вместе с подпиской на статус разработчика.

Для решения проблемы хранения пользовательских данных в облаке, можно выбрать одно из следующих направлений: развернуть собственную инфраструктуру или воспользоваться сервисом iCloud.

Разработка собственной инфраструктуры потребует поиска и покупки копии лицензионного ПО, предназначенного для хранения данных в БД, а также аренды или установки собственных серверов, которые необходимо расположить в разных частях света, и настроить репликацию между ними. Задача не из простых, тем более что требуемый продукт может требовать достаточно простой функционал и от БД нужно будет только сохранять и возвращать данные.

В случае если будет решено использовать iCloud, то разработчику не нужно заниматься развертыванием серверов, постоянным мониторингом системы и оплатой серверного оборудования – всеми этими задачами будет заниматься Apple. iCloud универсальная система, которая поддерживает все устройства, работающие на операционных системах iOS и macOS. Схема поддерживаемых сервисом iCloud устройств представлена на рисунке 1.

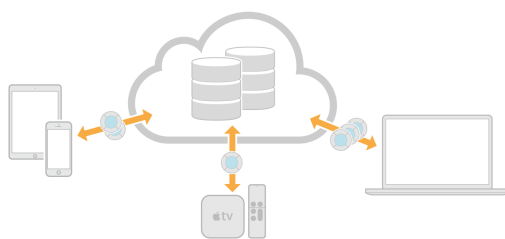


Рисунок 1. – Схема поддерживаемых сервисом iCloud устройств

Однако не стоит рассчитывать, что бесплатное решение от Apple будет во всем лучше собственной разработки. CloudKit имеет ряд жестких ограничений, которые могут быть критичны для многих приложений. Среди ограничений:

- 100 Мб дискового пространства;
- 2 Гб трафика;
- 40 запросов в секунду [1].

Данные ограничения с легкостью снимаются платной подпиской, но даже этого достаточно для большинства приложений, что делает данную технологию весьма полезной при ограниченном бюджете.

Практически любая современная компьютерная игра построена на взаимодействии между пользователями. Благодаря программистам из Apple, у iOS разработчиков есть возможность быстро и легко

оживить свою игру, добавив в нее элементы социальности и позволив сравнивать свои результаты с другими игроками или даже играть против них в реальном времени. Проанализируем три такие технологии.

Для реализации функционала таблицы рекордов игрока при помощи GameCenter от Apple, требуется создать таблицу в панели разработчика и одной строчкой кода отправлять результаты на сервер. Еще одной строчкой кода можно открыть таблицу или получить данные для отображения в собственном интерфейсе. Таблицы рекордов это один из самых мощных и действенных элементов социальности и разработка Apple значительно облегчает работу с большим количеством данных [2]. Пример таблицы рекордов представлен на рисунке 2.

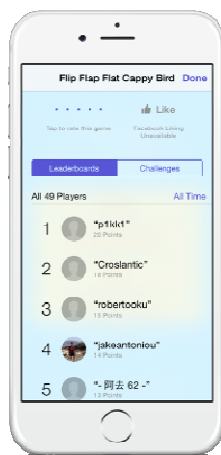


Рисунок 2. – Пример таблицы рекордов

Механизм достижений игроков, аналогично таблице рекордов, достаточно реализован Apple и доступен в панели управления игрой iTunesConnect. Разработчику необходимо создать достижение и вызвать событие его получения из игры в нужный момент времени. Сколько бы это заняло времени разрабатывая это с нуля, сложно представить, но Apple это сделали так, что практически все самые крупные разработчики мобильных игр отказываются от реализации собственного функционала в сторону встроенного сервиса достижений.

Одним из самых важных и сложных подсистем GameKit, является система игры «Игрок против игрока». Apple бесплатно предоставляет мощный функционал, на который многие компании тратят десятки тысяч долларов. При помощи данного сервиса, игроки могут играть против друг друга с разных устройств находясь в разных частях света, при этом имея минимальную задержку на передачу данных. Это один из самых спорных функционалов и многие предпочитают разворачивать свое собственное серверное решение, так как оно должно быть максимально гибким для разработки сложных и безопасных систем.

Проанализировав встроенные сервисы Apple, можно сделать **вывод**, что игровые сервисы позволяют реализовать базовые потребности разработчиков в необходимости внедрения функционала для социализации игрового процесса. Разработчики без труда могут реализовать игру против другого игрока, достижения и таблицы рекордов, а также позволяют хранить данные на удаленных серверах. Данный функционал предоставляется в рамках подписки разработчика и не требует дополнительных финансовых вложений, однако обладает не высокой гибкостью и не богатым функционалом. Собственные решения позволяют достичь большего уровня гибкости системы, однако их разработка и развертывание обходится в десятки раз дороже, чем встроенные сервисы Apple.

Таким образом, принято решение в дальнейшем использовать встроенные сервисы Apple в небольших и не сложных проектах, так как они позволяют значительно снизить стоимость разработки благодаря высокой скорости разработки.

ЛИТЕРАТУРА

1. CloudKit – AppleDeveloperPortal [Электронный ресурс]. – Режим доступа: <https://developer.apple.com/icloud/cloudkit/>. – Дата доступа: 24.09.2017.
2. Leaderboard – UnionAssets [Электронный ресурс]. – Режим доступа: <https://unionassets.com/iosnative/leaderboards-45/>. – Дата доступа: 25.09.2017.

УДК 004.42

**УДАЛЕННАЯ НАСТРОЙКА ПАРАМЕТРОВ ПРИЛОЖЕНИЯ
СРАВНЕНИЕ СОБСТВЕННОГО РЕШЕНИЯ И FIREBASE REMOTE CONFIG****А.И. ВАЛЬКОВИЧ***(Представлено: канд. техн. наук И.Б. БУРАЧЕНОК)*

Проведен анализ требований к системам удаленной настройки приложений. Осуществлен сравнительный анализ FirebaseRemoteConfig и собственного решения для удаленного изменения параметров приложения без перекомпиляции бинарного файла. Также автор делает о целесообразности собственной разработки системы удаленной настройки.

Несмотря на то, что разработкой приложений занимаются большие и опытные команды, нельзя всегда с уверенностью спрогнозировать поведение людей во время использования. Часто отклонения в балансе или ошибки в расчетах становятся видны только после запуска программного продукта на глобальный рынок, в тот самый момент, когда за ограниченный промежуток времени необходимо собрать как можно больше данных. Для успешного и грамотного управления современным программным обеспечением, необходимо заранее продумать и реализовать весь необходимый набор функционала, который позволит быстро менять параметры и настройки внутри приложения без его перекомпиляции.

В последнее время хорошей и оправданной практикой в разработке игровых приложений стало выносить как можно большее число настроек в облако. Примером таких настроек являются:

- цены во внутреннем магазине;
- время таймеров;
- наборы наград;
- локализация;
- настройки игровых уровней и др.

Быстрый доступ к данным настройкам позволяет экстренно вносить правки на основании аналитики пользовательских данных, что положительно сказывается всех показателях приложения и экономит разработчикам много времени на перекомпиляциях и загрузках приложения в AppStore.

На данный момент существует большое количество систем для удаленной настройки приложений. Большая часть из них платные, но существуют и бесплатные продукты. Среди бесплатных выделен сервис FirebaseRemoteConfig от Google. Логотип Firebase представлен на рисунке 1.

**Рисунок 1. – Логотип Firebase**

FirebaseRemoteConfig – облачная служба, которая позволяет менять поведение и внешний вид приложения, не требуя от пользователей загрузки обновления. При использовании RemoteConfig создаются значения по умолчанию, которые контролируют поведение и внешний вид приложения. Затем используется консоль Firebase для переопределения значений по умолчанию в приложении для всех пользователей или для сегментов пользовательской базы [1].

Выгодной особенностью Firebase в сравнении с остальными подобными сервисами, является возможность делить пользователей на сегменты и менять параметры отдельно для каждой группы пользователей. Это позволяет провести анализ, проанализировать результаты и принять решение об успешности эксперимента по изменению параметров. Как правило, пользователи делятся на группы на основании следующих признаков:

- используемая операционная система;
- язык устройства;
- страна проживания;
- случайный процент выборки и др.

Кабинет администратора приложения FirebaseRemoteConfig представляет собой таблицу, состоящую из двух колонок: параметра и значения.

Интерфейс панели администрирования FirebaseRemoteConfig представлен на рисунке 2.

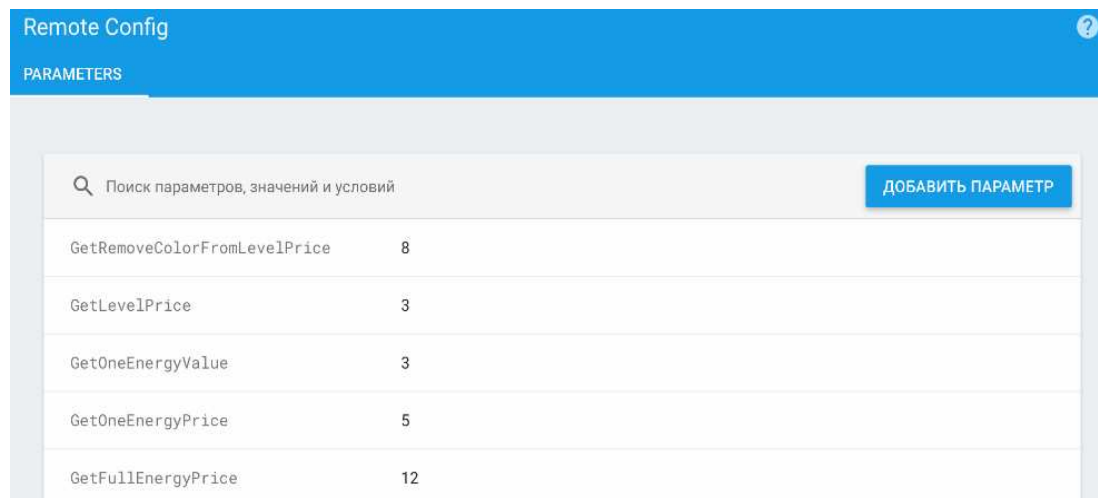


Рисунок 2. – Интерфейс панели администрирования FirebaseRemoteConfig

Как видно из рисунка 1, в приложении заданы 5 параметров, которым соответствуют 5 числовых значений. Изменение одного из них на сайте приведет к изменению параметров приложения при следующем его запуске.

Из минусов Firebase стоит выделить невозможность работы с массивами данных. На данный момент есть возможность сохранять только одиночные ключи, что усложняет работу с большим количеством данных. Еще одним минусом является не полная поддержка редактора Unity. Удаленный конфигурактор не работает в режиме разработки.

Альтернативой FirebaseRemoteConfig может стать собственное решение. Проанализировав минусы FirebaseRemoteConfig можно составить список функционала, который должен быть добавлен в систему для удобного ее использования:

- возможность сохранения массивов данных;
- возможность использования в редакторе unity.

Финальным вариантом файла конфигурации решено использовать файл XML, в котором описаны необходимые для игры параметры. Файл конфигурации приложения представлен на рисунке 3.

```

▼<root>
  ▼<ios>
    <refresh>3</refresh>
    <fail>2</fail>
    <level>1</level>
    <start>0</start>
    <refreshBattle>3</refreshBattle>
    <finishBattle>3</finishBattle>
    <dailyModeRetry>2</dailyModeRetry>
    <dailyModeRefresh>2</dailyModeRefresh>
    <onlineMenu>1</onlineMenu>
    <showTutorialText>1</showTutorialText>
    <dailyAttempts>3</dailyAttempts>
    <oneAttemptPrice>100</oneAttemptPrice>
    <pvpRewardCoins>15</pvpRewardCoins>
    <pvpRewardWins>5</pvpRewardWins>
    <pvpRewardPrice>15</pvpRewardPrice>
    <pvpRewardIfWin>25</pvpRewardIfWin>
    <pvpBigRewardPrice>150</pvpBigRewardPrice>
    <pvpBigRewardIfWin>250</pvpBigRewardIfWin>
  </ios>
  <localization></localization>
</root>

```

Рисунок 3. – Файл конфигурации приложения

Конфигурационный файл загружается с сервера в момент запуска игры, и внутриигровые параметры синхронизируются с пришедшими из облака актуальными значениями.

Данный подход является более гибким, но менее безопасным. Поддержка своей инфраструктуры требует дополнительных затрат и постоянного мониторинга состояния сервера.

Проанализировав 2 системы, можно сделать **вывод**, что у каждой системы есть свои плюсы и минусы. Собственная реализация удаленной системы конфигурации может в перспективе оказаться выгодной разработкой. Имеющиеся на данный момент аналоги не обладают достаточным функционалом и не имеют возможности расширения, что делает их непригодными для большого списка задач, однако они не требуют дополнительных финансовых вложений и обладают высокой стабильностью работы.

Заключение

Исходя из результатов проведенного анализа, принято решение – использовать в текущих проектах свою собственную систему удаленной настройки до тех пор, пока в популярные аналоги не добавят необходимый для более гибкой настройки функционал.

ЛИТЕРАТУРА

1. FirebaseRemoteConfig [Электронный ресурс]. – Режим доступа: <https://firebase.google.com/docs/remote-config/>. – Дата доступа: 26.09.2017.

УДК 004.946

ОБЗОР СРЕДСТВ СОЗДАНИЯ СИСТЕМЫ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ

Г.В. ЕМЕЛЬЯНОВ

(Представлено: канд. техн. наук И.Б. БУРАЧЕНОК)

Представлена схема работы программно-аппаратных компонентов, взаимосвязи логических модулей для достижения эффекта дополненной реальности, с использованием различных технических средств. Осуществлен обзор средств создания системы дополненной реальности в различных сферах человеческой деятельности.

В современных источниках можно встретить терминологию дополненной реальности, представленную в виде сокращения (аббревиатуры) AR или Augmented Reality. Эта технология позволяет существенно расширить область данных воспринимаемых человеком. Такое расширение сознания достигается за счет переноса в реальный мир цифровой информации. Процесс формирования дополненной реальности происходит при помощи камеры смартфона, веб-камеры или прочего устройства, которое может обрабатывать видеосигнал. Специальная программа дополнит картинку необходимыми виртуальными объектами. В качестве элементов AR-технологии могут выступать видео и аудио материалы, 3D-модели, а также текстовый контент.

Основным различием между технологией дополненной реальности и виртуальным аналогом является соотношение информации, полученной из реального мира и обработанной компьютером. Виртуальная реальность пытается всецело поглотить настоящий мир, а дополненная лишь расширяет понимание о происходящем в нем процессах.

Принцип создания дополненной реальности. Принцип создания дополненной реальности определяется техническими возможностями, с помощью которых она реализуется.

Средства создания дополненной реальности следующие:

- мобильные устройства (планшеты, смартфоны);
- очки дополненной реальности;
- телевизор (или экран компьютера);
- игровой компьютер типа Kinect;
- средства для воспроизведения AR в открытом пространстве;
- различные тренажеры;
- медицинское оборудование.

Общую схему создания дополненной реальности можно описать следующим образом: оптический сканер (камера) устройства считывает (снимает) изображение реального объекта; программное обеспечение устройства проводит идентификацию и анализ полученного изображения, выбирает или вычисляет соответствующее изображению видимое дополнение, объединяет реальное изображение с его дополнением и выводит итоговое изображение на устройство визуализации. Так получается дополненная реальность. Логика работы описанной схемы представлена на рисунке 1.

Далее подробно рассмотрим наиболее распространенные средства создания дополненной реальности.

Мобильные устройства. Практическое применение их в области дополненной реальности – это, как правило, развлекательные или рекламные цели. Камера планшета (мобильного телефона) передает снимаемое изображение в специальную программу, которую пользователь предварительно скачивает в устройство. При анализе изображения программа распознает это изображение, выбирает связанное с изображением содержимое (это может быть текст, видеоролик, 3D-модель) и воспроизводит его на фоне изображения с камеры. Например, при наведении камеры телефона на рекламу в журнале вы видите на экране телефона объемное изображение товара, который продвигает эта реклама. В некоторые программы воспроизведения такой дополненной реальности заложены функции управления этим объектом (масштабированием, поворотом, вызовом звуковых сигналов при касании разных частей объекта). Возможности визуализации дополненной реальности определяются только фантазией разработчиков программы.

Очки дополненной реальности. Это просто очки-компьютер. Человек, надевший на себя такие очки, видит через прозрачные линзы реальный мир и одновременно наложенные на него виртуальные объекты. В очках дополненной реальности встроена система, отслеживающая движение глаз, а программное обеспечение, которое суммирует виртуальную и реальную картинку, позволяет создавать иллюзию обзора панорамы. Например, в реальный городской пейзаж можно встроить старинные замки,

и в зависимости от поворота головы (взгляда) человек видит разные участки такого дополненного пейзажа. Примером таких очков является устройство Microsoft HoloLens [1].

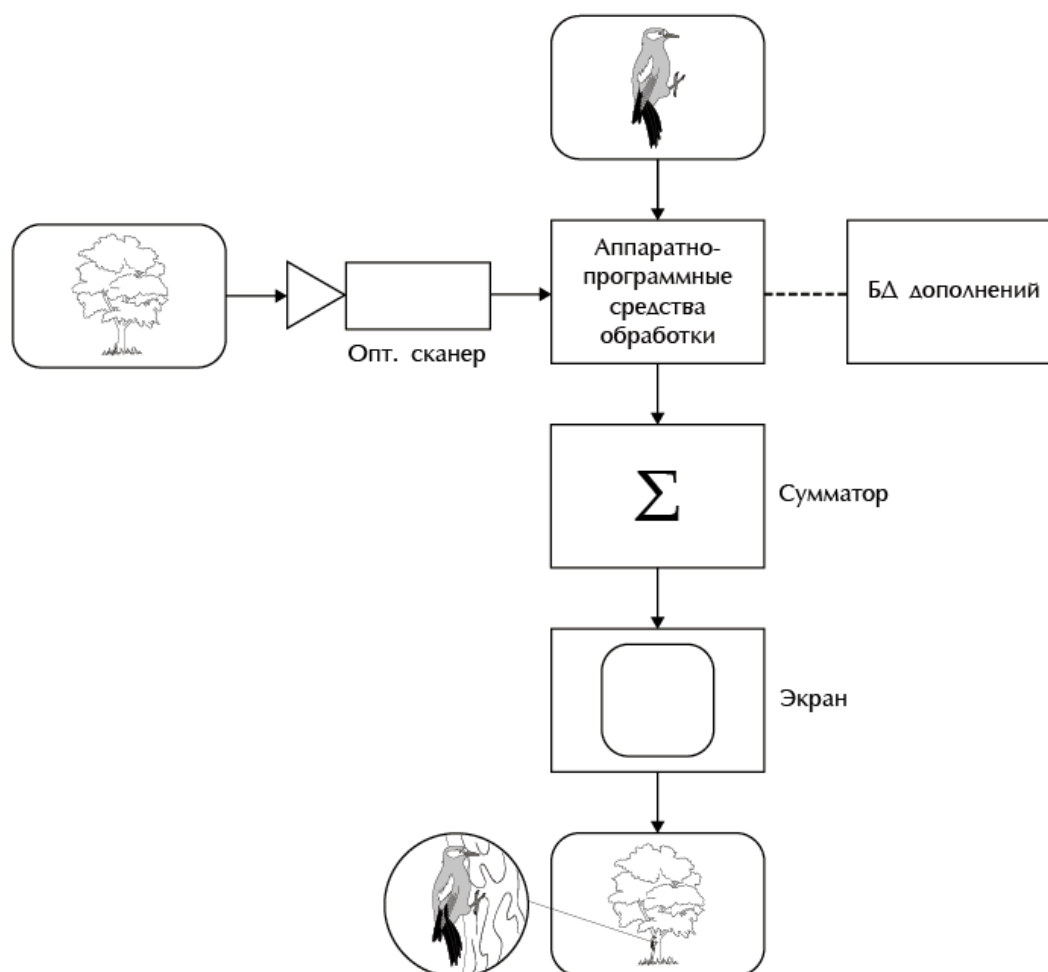


Рисунок 1. – Логика работы системы дополненной реальности

Экран компьютера. Обычно применяется для показа телетрансляции, например, спортивных матчей. Видеосигнал, прежде чем отправиться в эфир, пропускается через специальное программное обеспечение, которое накладывает на картинку какую-либо видеoinформацию. Телезритель видит уже измененный видеосигнал, содержащий в себе реальную и компьютерную картинку. Как правило, он даже не осознает, что видит дополненную реальность. Очевидно, что телесигнал в эфире уходит с некоторой задержкой, необходимой для компьютерной обработки сигнала. Такой прием часто применяется во время спортивных матчей для нанесения на реальную картинку цифровых отметок и линий, показывающих расстояние до ворот, наличие офсайда и т. п.

Игровой компьютер типа Kinect. Kinect – это игровой контроллер, работающий вместе с компьютером или игровой приставкой [2]. В состав контроллера входит видеокамера, которая позволяет программному обеспечению анализировать окружающее пространство, положение и жесты игрока. В состав оборудования также входят два инфракрасных датчика глубины и микрофон. Программное обеспечение получает трехмерное изображение и калибрует датчики с учетом уровня освещенности и окружающих условий, проводит распознавание мимики, голоса, движения тела. Таким образом, в игровой процесс, то есть в видеоизображение, формируемое игрой, встраивается сам игрок (и даже несколько игроков) и окружающее его пространство.

Воспроизведение AR в открытом пространстве. Для того чтобы формировать дополненную реальность в открытом пространстве, нужны видеокамеры, компьютер со специальным программным обеспечением, лазерная установка. Видеокамера получает и пересылает в компьютер реальное изображение, программное обеспечение производит анализ изображения, лазерная установка световым лучом

прорисовывает на реальном изображении дополнительное изображение. Так, например, работает бильярдный тренажер [3]. Игрок целится кием по шару на бильярдном столе, программа анализирует положение шаров, кия, игрока, вычисляет предполагаемую траекторию движения шара. Лазерная установка проецирует изображения траектории прямо на бильярдный стол. Эту траекторию видит игрок и может корректировать положение кия. Получается классическая система управления с отрицательной обратной связью.

Специальные средства AR. К ним относятся всевозможные специализированные технические устройства, отличающиеся по способу взаимодействия с человеком и способу отображения информации. Хотя общий принцип сложения (синтеза) реального и мнимого изображений с помощью компьютерной программы остается неизменным.

К **специальным средствам** относятся специализированные шлемы пилотов [4] для тренажеров или реальных условий полета. На стекло шлема выводится некая информация поверх окружающей обстановки, такой шлем отслеживает положение глазных яблок пилота.

Также существует специальное медицинское оборудование, применяемое для обучения врачей или для реального проведения операций. В первом случае, например, проводится обучение медика поиску кровеносного сосуда и введению в него иглы. Игла реальная – кровеносный сосуд виртуальный, но через сложную систему гидравлических механизмов врач ощущает его как реальный (он даже может почувствовать пульс), а при точном попадании иглы в сосуд, на экране устройства из сосуда выходит кровь. Во втором случае, например, при эндоскопических операциях, на изображение оперируемого органа, выводимое на экран, накладывается рентгеновское изображение опухоли, полученное ранее. Это позволяет врачу с большей точностью локализовать опухоль внутри органа.

Заключение

Автором представлена схема работы программно-аппаратных компонентов, взаимосвязи логических модулей для достижения эффекта дополненной реальности, с использованием различных технических средств. Также представлены примеры использования в различных сферах человеческой деятельности. Наиболее доступным и массово распространенным является использование мобильных устройств в качестве средств для реализации системы дополненной реальности. Данные устройства обладают необходимыми техническими характеристиками для создания подобных систем.

ЛИТЕРАТУРА

1. Hololense [Электронный ресурс]. – Режим доступа: <http://hololenses.ru/obzory-i-stati/obzor-microsoft-hololens.html>. – Дата обращения: 23.09.2017.
2. Kinect [Электронный ресурс]. – Режим доступа: <https://roscontrol.com/product/xbox-360-e-500gb-kinect>. – Дата обращения: 23.09.2017.
3. Бильярдный тренажер [Электронный ресурс]. – Режим доступа: <http://7days.kiev.ua/2013-03-10/bilyardnyj-trenazher-ili-dopolnennaya-realnost-v-dejstvii.html>. Дата обращения 23.09.2017.
4. Шлем для пилотов [Электронный ресурс]. – Режим доступа: <http://fotostrana.ru/public/post/335697/627210663/>. – Дата обращения: 23.09.2017.

УДК 004.946

**ПРИНЦИПЫ ПРОЕКТИРОВАНИЯ ГРАФИЧЕСКИХ ОБЪЕКТОВ
ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ****Г.В. ЕМЕЛЬЯНОВ***(Представлено: канд. техн. наук И.Б. БУРАЧЕНОК)*

Представлен обзор подходов базовых принципов создания интерфейса виртуальной реальности, основанный на психологии человеческого восприятия объектов окружающей среды.

Развитие технологии виртуальной реальности (VR) набирает обороты, однако по-прежнему ощущается нехватка методик и рекомендаций по дизайну. В данной работе подробно остановимся на пользовательском интерфейсе и проблемах восприятия, создаваемых технологиями виртуальной реальности.

Исторически сложилось так, что цифровые интерфейсы создаются из расчета соответствия характеристикам двумерных дисплеев. Дизайнерам приходится вписывать содержимое и навигацию в рамки экранов, преобразуя опыт взаимодействия с реальным миром в иконки и прочие элементы интерфейса. Но формирование виртуальной среды из двумерных элементов разрушает эффект погружения, необходимый для VR. Поэтому дизайн в этой сфере подразумевает не перенос привычных двумерных методик в трехмерное окружение, а поиск новой парадигмы.

В основе работы лежит литература, исследующая человеческое восприятие окружающей среды. Хотя основной массив работ по экологической психологии был написан в 1970-е годы, все эти знания полностью применимы и к виртуальной реальности. Дизайнерам нужно расширять свои знания по разным дисциплинам: архитектуре, звуковой и световой сценографии, а также физике, чтобы иметь возможность создавать полностью контролируемый опыт взаимодействия, формируя VR так, чтобы направлять пользователей как это нужно разработчикам.

Решением для разных проблем на примере нейтрального VR-окружения, в форме разных характерных ситуаций. Эти ситуации охватывают лишь часть широкого разнообразия новых видов взаимодействий и решений, предлагаемых VR. Будущее этих сред зависит от создаваемого для них контента. Сегодня VR-продукты чаще всего симулируют реальную жизнь. Но когда эффект новизны пройдет, контент будет адаптирован под особенности среды.

Далее будем агрегировать дизайн-решения применимые к виртуальной реальности.

Дизайн-решения в VR

Роль земли. Когда самолет летит в облаках, то снаружи ничего не видно, и это может сильно дезориентировать. А когда облака кончаются, и пилот может видеть землю, небо и горизонт, снова появляется чувство ориентации в пространстве [1]. При плохом дизайне опыта взаимодействия в VR-пользователи тоже могут испытывать дезориентацию, что неизбежно приводит к укачиванию. Видеть землю и горизонт в VR не менее важно, чем и в реальном мире. Схематическое отображение земли представлено на рисунке 1.

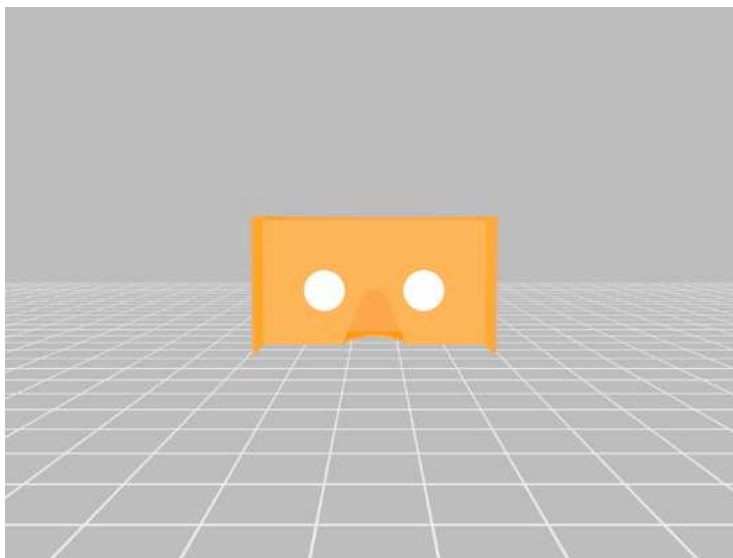


Рисунок 1. – Схематическое отображение земли

Атмосфера. Атмосферная (воздушная) перспектива может помочь пользователям понять масштаб виртуального окружения, что добавляет реализма. Чем дальше объект, тем больше воздуха и препятствий находится между объектом и наблюдателем, поэтому далеко расположенный объект видится менее отчетливо, чем находящийся ближе [2]. Это постепенное увядание ландшафта необходимо для того, чтобы дать пользователю представление о глубине и расстоянии.

Особенности рельефа. Выделяется восемь основных свойств рельефа.

1. *Земная поверхность* редко представляет собой открытое ровное пространство. Обычно она чем-то загромождена. Открытое пространство позволяет двигаться по земле в любом направлении, а загроможденное – только в проходы.

2. *Путь* – обеспечивает перемещение от одного места к другому, между другими свойствами рельефа.

3. *Препятствие* – объект, с которым можно столкнуться.

4. *Барьер* – особый вид препятствия, обычно блокирующий поле зрения и движение.

5. *Водная преграда* – не позволяет передвигаться пешком.

6. *Край обрыва* ограничивает возможность приблизиться к какому-то объекту. Это опасная зона, сухопутные животные избегают таких мест.

7. *Шаг* – это единица измерения, которую можно использовать для определения продолжительности спуска или подъема.

8. От покатости зависит возможность пешего перемещения, он определяется углом текстурой поверхности.

Используя эти свойства в качестве строительных блоков при создании виртуального окружения, возможно создавать интуитивный опыт взаимодействия.

Влияние звукового окружения. В VR не нужно прилагать каких-то усилий для смены окружающего пространства, хотя такие неожиданные и ошеломляющие перемены могут сбивать пользователя с толку и вызывать тошноту. Поэтому желательно аккуратно подводить его к моменту изменения окружения, предварительно плавно усиливая звуковой фон, характерный для нового места, а потом уже показывая само место. Звуковое оформление подготавливает мозг человека, что смягчает фактор неожиданности.

Управление пользователем с помощью объектов. В загроможденном пространстве бывает трудно вести пользователя по определенному пути без традиционных навигационных элементов интерфейса. Но это может разрушить эффект погружения. Зато можно направлять мягко и ненавязчиво, с помощью небольших изменений в окружающем мире.

Прицел, зависящий от контекста. В неотслеживаемом VR (например, Google Cardboard [3]) прицелы используются для отображения линии взгляда пользователя. Изображение точки фокусировки помогает ориентироваться в пространстве, а также упрощает перемещение и взаимодействие с объектами.

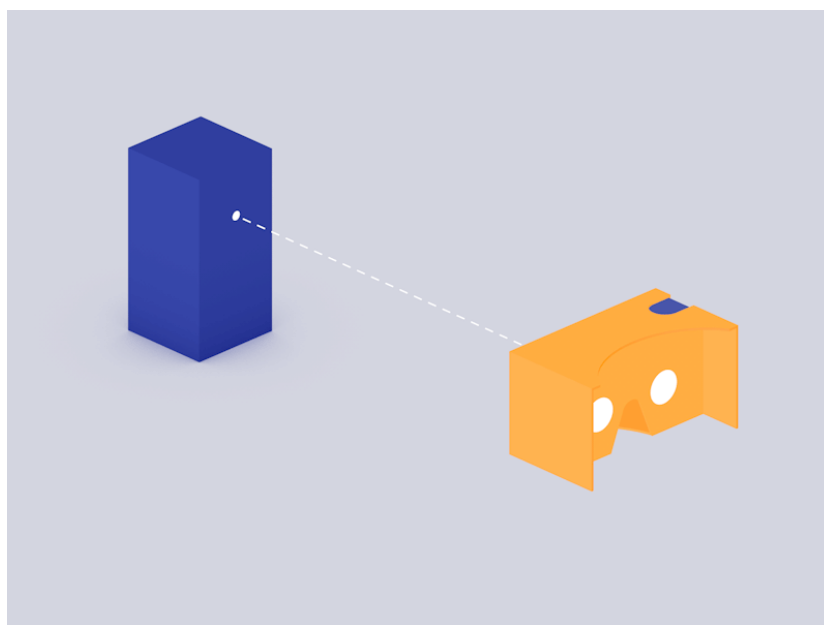


Рисунок 2. – Схематическое отображение прицела

Различные задачи требуют разной реакции от прицела.

- *Состояние покоя.* Прицел должен быть минимального размера, подсказывая лишь расположение центра кадра.

- *Движение.* Прицел активируется, когда пользователь смотрит на любой объект, к которому можно приблизиться. В этом случае прицел превращается в более крупный указатель, активирующий подсветку выделяемой зоны с помощью круга, перемещающегося в зависимости от точки зрения пользователя, как показано на рисунке 2.

- *Взаимодействие с объектами.* Когда пользователь уделяет внимание объекту, с которым можно взаимодействовать, то прицел должен на это среагировать.

- Цвет прицела должен адаптироваться к уровню окружающей яркости, переключаясь между темным и светлым режимами, чтобы быть одинаково заметным при любом освещении.

- Объект в качестве прицела. Замена прицела каким-то трехмерным объектом может быть подсказкой для взаимодействия (например, прицел в виде ключа говорит о том, что нужно открыть замок).

Интерактивные объекты. Если не с каждым объектом можно взаимодействовать, то пользователю нужно давать подсказку, какие же объекты являются интерактивными. Обычно полезен контекстуальный прицел, но иногда, чтобы избежать путаницы, сами интерактивные объекты тоже должны изменяться. Изменение может быть очень небольшим, например, при взгляде на объект он может затеняться или издавать слабый звук, описывающий его поведение (например, выключатель света может щелкать).

Заключение

В представленной работе разработана методология подходов к разработке дизайна для VR. На данный момент – это новая развивающаяся среда, потенциально способная оказать на общество эффект. Заложены фундаменты в разработке интерфейсов и опыта взаимодействия для этой среды без того, чтобы по умолчанию применять уже существующие решения.

ЛИТЕРАТУРА

1. Gibson, JamesJ. TheEcologicalApproachtoVisualPerception [Электронныйресурс]. – Режим доступа: https://daughtersofchaos.files.wordpress.com/2014/05/gibson_occluding-edge_1979.pdf. – Дата обращения: 23.09.2017.
2. Goldstein, E. Bruce. SensationandPerception [Электронный ресурс]. – Режим доступа: <http://zhenilo.narod.ru/main/students/Goldstein.pdf>. – Дата обращения: 23.09.2017.
3. GoogleInc. Cardboard [Электронный ресурс]. – Режим доступа<https://ustwo.com/work/google-cardboard>. – Дата обращения 23.09.2017.

УДК 004.855.5

РАЗРАБОТКА НЕЙРОСЕТИ ДЛЯ УПРАВЛЕНИЯ БЕСПИЛОТНЫМ АППАРАТОМ

М.Ю. ВАШКЕВИЧ

(Представлено: Д.В. ПЯТКИН)

Представлен практический способ создания нейронной сети для управления беспилотным аппаратом. Также рассмотрены основные моменты создания и обучения нейронной сети. Задача реализации интерфейса решалась с использованием JAVA.

Используем нейронную сеть для того, чтобы транспортное средство управляло собой само, избегая препятствий. Добиваемся этого путем выбора соответствующих входов/выходов и тщательного обучения нейронной сети. Вначале передаем сети расстояния до ближайших препятствий вокруг аппарата, имитируя зрение водителя-человека. На выходе получаем ускорение и поворот руля транспортного средства. Также необходимо обучить сеть на множестве стратегий ввода-вывода. Результат впечатляющий даже с использованием всего лишь нескольких нейронов. Беспилотный аппарат двигался, обходя препятствия, но возможно сделать некоторые модификации, чтобы это программное средство справлялось с более специфическими задачами [1].

Принцип разработанной нейронной сети

Нейросеть можно сравнить с функцией. Есть множество входов. На вход подается массив препятствий. Эти данные обрабатываются нейросетью как функцией и реакция нейронов является выходом из функции. Функция $f(x) = y$ преобразует значение x (одно измерение) в y (одно измерение). Функцией, которая будет преобразовывать значения и есть нейронная сеть, а более конкретно – нейронная сеть обратного распространения. Использоваться будет нейронная сеть обратного распространения, так как данная нейронная сеть выполняет работу быстрее всего при нескольких входах выходах. Поскольку мы должны работать с несколькими входами и выходами, то данная система наиболее подходящая.

Когда нейронная сеть состоит всего из нескольких нейронов, мы можем вычислить веса, необходимые для получения приемлемого результата. Но по мере увеличения числа нейронов, увеличивается и сложность вычислений. Сеть обратного распространения можно обучить, что установит необходимые веса. Мы просто должны предоставить искомые результаты с соответствующими им входами.

После обучения нейронная сеть будет реагировать – выдавать результат, близкий к желаемому, при подаче известного результата, получать правильный ответ при любом входе, не соответствующем обучающему [2].

Для обучения нейросети будем использовать обратное распространение ошибки, а также обучение с учителем. Для обучения с учителем, нейросети необходимо подать на вход возможные комбинации входов выходов. После обучения нейросеть сможет принимать решения согласно тому, какие данные она получила. Этот тип обучения был использован в связи с тем, что он дает наименьшую погрешность при вычислениях и при обучении нейросети.

Добавление нейронов: Имеем входной слой I с i нейронами и результирующий слой O с o нейронами. Мы хотим добавить один нейрон в средний слой M . Число соединений между нейронами, которые мы добавляем, равно $(i+o)$.

Добавление слоев: Входной слой содержит I с i нейронами и результирующий слой O с o нейронами. Необходимо добавить M слоев с m нейронов в каждом. Число соединений между нейронами, которые добавляются, равно $(m*(i+o))$.

После рассмотрения работы нейронной сети необходимо понять, как определить входы и выходы нейронной сети. Нейронная сеть сама по себе ничего не делает, если ей не передать информацию из виртуального мира и не подать ответ сети контроллеру транспортного средства.

На ввод в нейросеть необходимо подать матрицу препятствий, эта матрица содержит обозначение, есть ли препятствие впереди движения или же нет. Матрица должна содержать 9 элементов, каждый из которых должен быть нулем или единицей, ноль обозначает, что в данной ячейке нет препятствий и туда можно двигаться, если же в ячейке стоит единица, то это означает что там присутствует препятствие и туда двигаться невозможно. Каждая ячейка обозначает область перед беспилотником, эти области делятся на 9 частей и нумеруются от верхнего левого края слева на право, сверху вниз [3].

На выход же из нейросети может поступить на выбор: либо углы поворота беспилотного аппарата или же номер ячейки, куда необходимо двигаться.

В реальности нейронная сеть будет выглядеть так, как показано на рисунке 1.

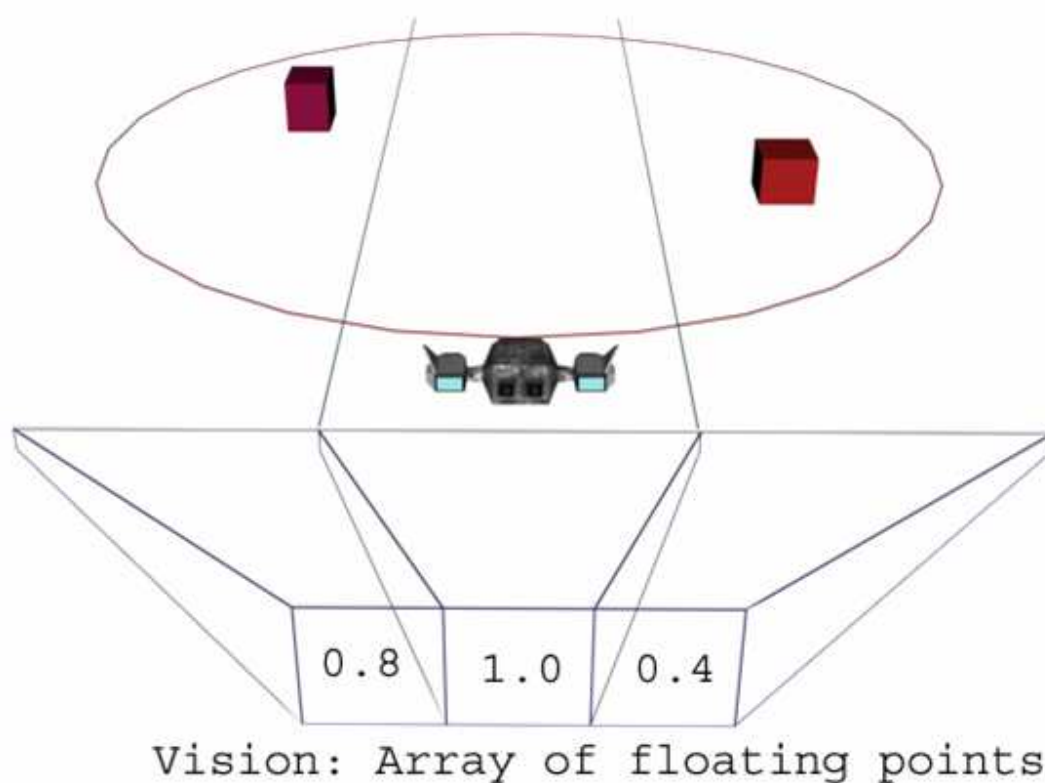


Рисунок 1. – Как система будет мир

Также чтобы вся система работала, было использовано следующее:

- сигмовидная функция (функция для того, чтобы приводить вычисления приближенно к промежутку от 0 до 1);
- метод обратного распространения ошибки (используется для обучения нейронной сети, чтобы нейронная сеть научилась выполнять поставленную задачу) [4].

Заключение

Результаты, полученные в ходе проведения исследования, позволили разработать нейронную сеть, которая в состоянии самостоятельно управлять беспилотником и обходить препятствия в сложной среде.

ЛИТЕРАТУРА

1. Нейросеть [Электронный ресурс]. – Режим доступа <https://geektimes.ru/post/101789/DES>. – Дата обращения: 27.09.2017.
2. Описание нейросетей [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/>. – Дата обращения: 27.09.2017.
3. Движение в плоскости [Электронный ресурс]. – Режим доступа <https://habrahabr.ru/post/322392/>. Дата обращения: 27.09.2017.
4. Описание нейросетей [Электронный ресурс]. – Режим доступа: http://elib.psuti.ru/Gorozhanina_Nejronnye_seti_uchebnoe_posobie.pdf. – Дата обращения: 27.09.2017.

УДК 004.838.5

НЕЙРОННЫЕ СЕТИ: ЧТО ОНИ МОГУТ СЕЙЧАС И ЧТО СМОГУТ В БУДУЩЕМ

М.Ю. ВАШКЕВИЧ**(Представлено: Д.В. ПЯТКИН)**

Рассматриваются современные и будущие возможности нейронных сетей. Сделан вывод, что нейронные сети не достигли пика своего развития, Но уже сейчас они могут выполнять многие действия, которые раньше мог делать только человек.

С первого момента создания компьютерачеловек пытался заставить его принимать решения самостоятельно. В качестве примера такой попытки можно привести тест Тьюринга, призванный прояснить ответ на вопрос, может ли компьютер проявить подобие интеллекта [1]. Нейронные сети – это проявление человеческого желания создать искусственный интеллект. Сейчас благодаря развитию техники, эта цель как никогда близка. Как пример можно рассмотреть самообучающиеся системы, которые помогают нам в интернете, такие как «Сири» от компании Apple.

Умные алгоритмы уже умеют находить и распознавать лица, делать поиск изображения в другом изображении, узнавать различные предметы. А нейронные сети пошли дальше и могут самостоятельно создавать произведения искусства.

Недавно Google в своем блоге опубликовали интересный способ использования нейронных сетей для создания картин. Каждая нейронная сеть обучается с помощью миллионов тренировочных картинок. Сеть имеет от 10 до 30 вложенных слоев с различными уровнями абстракции. Вначале картинка поступает на входной слой, который делает свою работу и передает информацию в следующий слой, пока на выходе не получится ожидаемый результат.

Важно понять, что именно происходит на каждом уровне системы. Каждый последующий слой извлекает новые черты изображения. Допустим, первый уровень определяет углы и ребра на картинке, второй – формы, и именно последние несколько слоев принимают решение о том, что изображено на картинке [2].

Чтобы нейронная сеть начала рисовать картины, на ее вход подается изображение и ставится задача – найти в нем определенную форму и утрировать ее. Пример рисования нейросети можно посмотреть на рисунке 1.

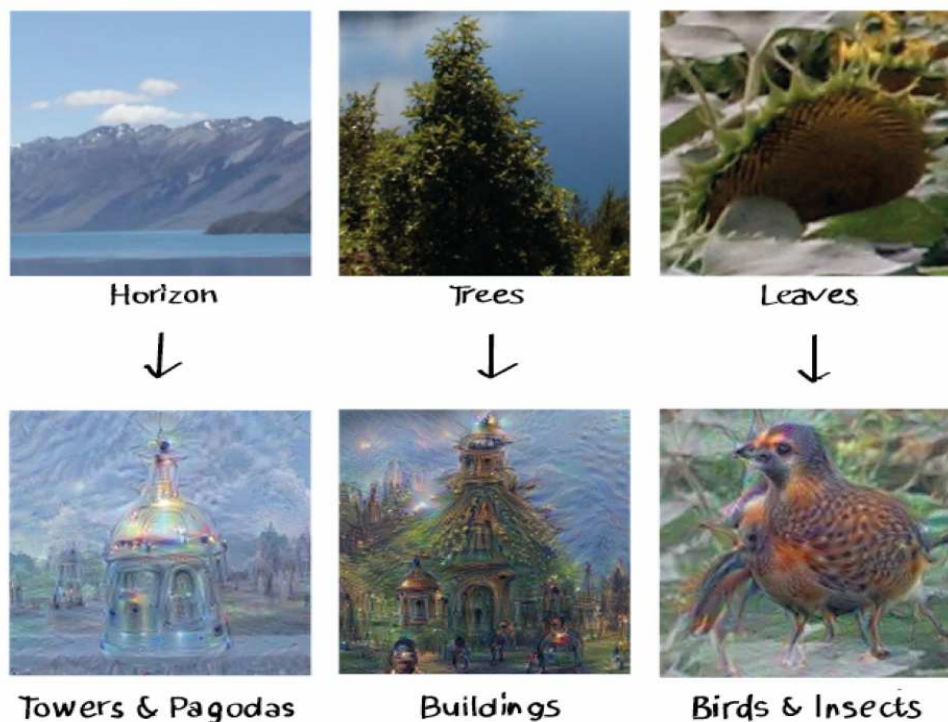


Рисунок 1. – Пример рисунков нейросети

Если нейронные сети уже умеют рисовать, то, скорее всего, рисованием это не закончится, и скоро появится информация о том, что нейронные сети научились создавать музыку или небольшие видео, или, что менее вероятно, они научатся писать книги.

Нейронные сети не достигли пика своего развития. Но уже сейчас они могут выполнять многие действия, которые раньше мог делать только человек. Однако если сравнить нейронную сеть с мозгом человека, то нейронной сети до нашего интеллекта еще очень и очень далеко. Самая крупная нейронная сеть состоит из 160 миллиардов нейронов и создана компанией DigitalReasoning, находится во Франклин-е, Теннесси. Так что уже сейчас создана нейронная сеть, которая содержит примерно в 2 раза больше нейронов, чем человеческий мозг. Скоро именно нейронные сети будут самыми разумными на нашей планете, ведь размер нейронов в нейронной сети можно увеличивать бесконечно, а человеческий мозг ограничен тем количеством, которое у него есть [3].

При первых попытках создать подобие человеческого мозга был придуман тест, чтобы определить, думает машина или просто повторяет заученное. Пока тест Тьюринга не пройден, нейронные сети не могут самостоятельно делать то, чему их не научили. Это говорит о том, что они лишь умные машины, способные решать поставленные перед ними задачи [4].

Заключение

Нейронные сети уже научились создавать и развиваться в том направлении, в котором их запрограммировали, и невозможно понять, что же еще они смогут сделать в будущем. Вполне возможно, что новое открытие будет сделано именно нейронной сетью.

ЛИТЕРАТУРА

1. Общие сведения о нейросети [Электронный ресурс]. – Режим доступа <https://ru.wikipedia.org/wiki/>. – Дата обращения: 27.09.2017.
2. Рисование нейросети [Электронный ресурс]. – Режим доступа <https://habrahabr.ru/company/io/blog/262267>. – Дата обращения: 27.09.2017.
3. Самая мощная нейросеть [Электронный ресурс]. – Режим доступа: <http://dic.academic.ru/dic.nsf/ruwiki/658474>. – Дата обращения: 27.09.2017.
4. Предположение к чему могут предти нейронные сети [Электронный ресурс]. – Режим доступа: <http://opencv.org>. – Дата обращения: 27.09.2017.

УДК004.582

**ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ-ОРГАНАЙЗЕРА
ПО УХОДУ ЗА КОМНАТНЫМИ РАСТЕНИЯМИ****К.П. ДИВИН****(Представлено: Д.В. ПЯТКИН)**

Рассматриваются основные особенности проектирования мобильного приложения по уходу за комнатными растениями. Приводятся определения и понятия; обзор средств разработки для Android.

Большое количество пользователей уже перестают пользоваться стационарными компьютерами, отдавая свое предпочтение мобильным средствам, которые позволяют быть на связи со всем миром 24/7. В отличие от стационарного компьютера мобильный телефон или планшет – это более личное устройство. В них хранятся информация о пользователях. Некоторые приложения уже способны помогать нам в планировании действий: стоит только запустить приложение и через некоторое время оно вычисляет местоположения офиса или дома и т.д. В статье пойдет речь об уходе за комнатными растениями посредством мобильного приложения, которое поможет пользователю организовать уход за растениями (полив, опрыскивание, подкормка, обрезка, пересадка, лечение болезней) и будет напоминать в заданное время, когда и что нужно совершить, также будет подробная информация по уходу за теми или иными растениями.

Комнатное растение – это растение в жилых или общественных помещениях, которые способны успешно выживать и развиваться в условиях помещения.

Мобильное приложение – программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах.

Приложения состоят из front- и back-end'a. Front-end – это компоненты программы, с которой взаимодействует клиент. Back-end или серверная часть – это компоненты, с которыми взаимодействует разработчик. Условно говоря, front-end находится на устройстве пользователя, а back-end на личном или арендованном сервере владельца программы.

Основные особенности мобильного приложения

Мобильное приложение по уходу за комнатными растениями поможет пользователю организовать уход за своими растениями и получить необходимую информацию о растениях.

Основные возможности приложения:

- 1) общие и индивидуальные напоминания;
- 2) создание, копирование, удаление растений;
- 3) отправка растений по почте друзьям и их импорт;
- 4) редактор частот напоминаний и фотогалерея;
- 5) карантин. Лечение растений;
- 5) гибкий поиск;
- 6) задачи на любую дату;
- 7) календарь с переходом в задачи на указанную дату;
- 8) просмотр и выполнение просроченных задач;
- 9) подробное описание растения и его фотография.

Обзор средств разработки для Android

Затрагивая самую обширную область в разработке, начнем с обзора средств для Android. Здесь в разработке программного обеспечения есть две основные IDE: проект Eclipse и разработка компании JetBrains, среда IntelliJ IDEA.

Eclipse в качестве инструмента сборки проекта выступает Ant. Данное IDE легко настраивается, интегрируется с необходимыми для разработки компонентами, такими как SDK Android, NDK (NativeDeveloperKit) и Java машиной. После успешной интеграции создать свой первый проект не составит труда. Уже после выбора нового проекта под Android автоматически создается пустой «HelloWorld» проект, который можно запустить на реальном девайсе. В качестве языка используется Java. Данная IDE очень проста в работе, поэтому новичкам в разработке советуют именно ее.

IntelliJ IDEA продукты JetBrains называют «УмнаяIntegrated development environment». Казалось бы, такие мелочи, как автоматическое закрытие скобок после условия, групповые перезаписи методов и автоматическое создание шаблонных классов (Interface, Singleton) на первый взгляд не очень важны. Тем не менее по статистике за год на одних только операциях 'rename' экономится порядка 120 часов ра-

бочего времени! Самая главная дилемма программистов – «как назвать переменную» – в данной Integrateddevelopmentenvironment решена. К примеру, если у вас есть базовый класс "Item", и вы создаете из данных элементов массив, то среда разработки автоматически предложит назвать их "Items".. Немаловажной является возможность изменения темы оформления. При длительной работе (8 часов в день) светлый фон намного сильнее утомляет глаз, чем темно-серый. Заметим, что данный проект не является открытым, однако команда разработчиков практически еженедельно выпускает обновления с исправлением ошибок.

Также стоит упомянуть и важный элемент в разработке графический редактор интерфейса. Возможность создавать интерфейс путем Drag-and-Drop намного упрощает задачу и ускоряет ее выполнение. Результат можно сразу увидеть на эмулируемом устройстве.

AndroidStudio в заключении следует упомянуть среду разработки AndroidStudio. Это результат сотрудничества JetBrains и Google, ориентированный на разработку приложений Android. По большому счету, это та же самая IDEA, за исключением того, что в качестве сборщика используется Gradle, а не Ant. Все изменения, которые вносятся в AndroidStudio через некоторое время, появляются в IDEA и наоборот. Google рекомендует именно AndroidStudio для разработки приложений.

Заключение

В ходе проведенного исследования рассмотрены одни из самых популярных на сегодняшний день средства для нативной разработки мобильных приложений и выбраны средства для разработки. Рассмотрены основные особенности мобильного приложения по уходу за комнатными растениями.

На данный момент ведутся работы по увеличению функциональности.

ЛИТЕРАТУРА

3. Eclipse (среда разработки) [Электронный ресурс]. – Режим доступа <https://ru.wikipedia.org/wiki/Eclipse>. – Дата обращения: 27.09.2017.
4. IntelliJIDEA [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/IntelliJ_IDEA. – Дата обращения: 27.09.2017.
5. AndroidStudio [Электронный ресурс]. – Режим доступа https://ru.wikipedia.org/wiki/Android_Studio. – Дата обращения: 27.09.2017.

УДК 004.582

**РАЗРАБОТКА ИНТЕРФЕЙСА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ОРГАНИЗАТОРА,
ПО УХОДУ ЗА КОМНАТНЫМИ РАСТЕНИЯМИ****К.П. ДИВИН****(Представлено: Д.В. ПЯТКИН)**

Рассматривается практический способ создания интерфейса для мобильного приложения по уходу за комнатными растениями. В качестве цели выступает написание мобильного приложения для пользователей, которые забывают о своевременном уходе за своими растениями, а также создание интуитивно понятного интерфейса.

Цель разработки интерфейса мобильного приложения по уходу за комнатными растениями – создание удобного для пользователя в обращении с приложением, а также комфортное его использования, с помощью которого будет легко ухаживать за растениями.

Принцип разработки интерфейса мобильного приложения

Опишем несколько приемов работы с приложениями, которые можно назвать эффективными и проверенными. В идеале приложение для мобильного устройства должно работать со скоростью мысли. Более того, интерфейс приложения должен быть понятен даже новичку.

Стоит выделить пять основных факторов, имеющих важное значение при разработке взаимодействия с приложением.

Целеориентированность. Создание интерфейса для конкретного пользователя. Сегодня в сети большое количество данных о различных категориях пользователей, причем многие материалы – исследования, обзоры – доступны бесплатно. Изучение этой информации поможет создать приложение, которое полностью соответствует потребностям целевой аудитории.

Юзабилити. Приложение должно быть удобным и интуитивно понятным. К примеру, если вы хотите указать ссылку для перехода на сторонний ресурс, то оформляйте ее привычным образом – с помощью подчеркнутого голубого текста. Удобство и практичность – это первый шаг на пути к тому, чтобы ваша программа стала желанной для пользователя.

Возможность (аффёрданс) и символичность. Аффёрданс – это функция. Для простоты снова воспользуемся приемом со ссылкой. Так, голубой подчеркнутый текст указывает на то, что клик по нему переведет пользователя по какому-то адресу. Подобные символы нужно использовать таким образом, чтобы пользователь не размышлял о том, что может означать тот или иной элемент интерфейса. Практичность и рациональность – наше все.

Обучаемость. В идеале пользователь должен без труда догадываться, как работать с программой. Здесь приходят на помощь знакомые и привычные схемы оформления приложения. Они должны помочь человеку без проблем привыкнуть к программе.

Фидбек и время ответа. Отклик приложения должен давать пользователю представление о том, выполнена задача или нет. Это может быть обычный звуковой сигнал или нечто более сложное – например, модальное окно. Убедитесь в том, что фидбек приложения соответствует положениям, установленным NielsenNormanGroup.

Первым шагом при создании целеориентированного интерфейса является изучение своей аудитории. Размер дисплея – это не единственное ограничение при разработке мобильного приложения. Пользователи также формируют требования к интерфейсу и необходимо их учитывать.

В этом вопросе есть четкая тактика, состоящая из трех положений:

Personas: помогает понять, что будет побуждать пользователя выполнить то или иное действие внутри приложения. Это формальные, теоретические модели реальных пользователей.

UserScenarios: обеспечивает моделирование различных ситуаций, помогает предсказать действия пользователя. Благодаря этому можно разработать интерфейс, который оптимально подходит для смоделированных пользователей и задач, которые они хотят выполнить.

Experiencemaps: здесь изучаются все возможные условия отдельного взаимодействия. Схема может описать каждый шаг пользователя, который будет выполнен с высокой вероятностью на определенном этапе работы с приложением. Такая схема поможет понять эмоции и обстоятельства, которые приводят к выполнению каждого действия.

Этот пункт позволяет параллельно проводить разработку приложения и изучать возможное поведение пользователей. Идеальным вариантом является набросок, по которому изучается взаимодействие пользователей с контентом – работу человека с содержимым программы можно проиллюстрировать на бумаге. Это поможет понять, как примерно будут вести себя пользователи внутри приложения.

Прежде чем сделать прототип, рукописная схема поможет изучить наиболее важную часть приложения – контент. Понимание возможной схемы взаимодействия пользователя с контентом поможет дать более точную оценку числа страниц/экранов, необходимых в программе.

Интерфейс мобильного приложения будет разработан с помощью средства разработки Eclipse и установленным на него плагином ADT, также нужно установить JDK и JRE, для тестирования и проверки работоспособности приложения будет использоваться симулятор или же виртуальная машина.

Мобильное приложение будет состоять из главной страницы, на которой размещены пункты, выбираемые пользователем, такие как растения, задачи, календарь и настройки и т.д. В пункте растения можно создавать, редактировать или удалять растения. При добавлении растений, нужно будет из списка предложенных выбрать нужное. После того как пользователь выбрал растение, выполняется переход на страницу с данным растением на которой будет основная информация о растении. Также будет возможность добавить фотографию данные растения и добавления напоминаний о поливе и т.д. В пункте задачи будет предоставлена информация о каждом из растений пользователя с их напоминанием о поливе или каких-либо других действий. В пункте календарь, будет отображаться обычный календарь, но если на определенный день добавлена какая-либо задача, то день с задачей будет отображаться по-иному и с количеством задач на день. В пункте настроек будет отображаться информация о приложении, текущее время, включены или выключены напоминания и т.д.

Заключение

Мобильное приложения по уходу за комнатными растениями – будет разработан удобный и интуитивно понятный интерфейс, который не будет требовать дополнительного обучения или контента на то направленное. На данный момент ведутся работы по увеличению функциональности.

ЛИТЕРАТУРА

1. Дизайн мобильных приложений [Электронный ресурс] – Режим доступа <https://habrahabr.ru/company/appodeal/blog/260095/>. – Дата обращения 27.09.2017.
2. Eclipse(среда разработки). [Электронный ресурс] – Режим доступа <https://ru.wikipedia.org/wiki/Eclipse>. – Дата обращения 27.09.2017.
3. IntelliJ IDEA. [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/IntelliJ_IDEA. – Дата обращения: 27.09.2017.
4. AndroidStudio [Электронный ресурс]. – Режим доступа https://ru.wikipedia.org/wiki/Android_Studio. – Дата обращения: 27.09.2017.

УДК 004.582

ИСПОЛЬЗОВАНИЕ SOCKET.IO ДЛЯ АСИНХРОННОГО И СИММЕТРИЧНОГО ОБМЕНА ДАННЫМИ

*М.А. БАЛАБАШ**(Представлено: Д.В. ПЯТКИН)*

Рассматривается javascript-библиотека с открытым исходным socket.io, при помощи которой можно осуществлять обмен данными между клиентом и сервером в реальном времени.

Интернет создавался главным образом на основе так называемой парадигмы запросов и ответов, реализованной с помощью протокола HTTP. Она заключается в том, что пока клиент не отправит запрос на открытие следующей страницы, она не загрузится. Примерно с 2005 года с появлением технологии AJAX работа в Интернете стала более динамичной. При этом обмен данными по протоколу HTTP все равно инициировался клиентом, и это требовало от пользователя выполнения определенных действий или периодического опроса сервера для загрузки обновленной информации.

Технологии, позволяющие серверу отправлять клиенту новые данные в момент их появления, существуют довольно давно. Они известны как Push или Comet. Один из распространенных способов создания иллюзии соединения, инициируемого сервером. Это так называемый *метод длинного опроса*. Его суть в том, что клиент создает подключение к HTTP-серверу и оно не закрывается до отправки ответа. Если на сервере действительно есть обновленные данные, он отправляет ответ (в других технологиях для этого используются Flash, multipart-запросы XHR и особые HTML-файлы). Технология длинного опроса, как и другие подобные приемы, прекрасно работает. Вы пользуетесь ими каждый день, например, в чате Gmail.

Однако у этого метода есть один недостаток – он использует протокол HTTP, что плохо подходит для приложений с низким временем реакции. В частности, имеем в виду многопользовательские браузерные игры от первого лица и другие сетевые игры в реальном времени.

Основная часть

В спецификации WebSocket определен API для создания сокетных соединений между браузером и сервером. Иными словами, между клиентом и сервером устанавливается постоянное подключение, в котором обе стороны могут инициировать обмен данными.

Socket.IO главным образом использует протокол WebSocket, но если нужно, использует другие методы, например Adobe Flash сокеты, JSONP запросы или AJAX запросы, предоставляя тот же самый интерфейс. Помимо того, что Socket.IO может быть использована как оболочка для WebSocket, она содержит много других функций, включая вещание на несколько сокетов, хранение данных, связанных с каждым клиентом, и асинхронный ввод/вывод.

Основными сферами применения технологии являются:

1. Быстрый обмен данными для онлайн игр, чатов и пр.
2. Веб-приложения с интенсивным обменом данными, требовательные к скорости обмена и каналу.
3. Push уведомления
4. IoT-приложения.
5. Удаленный доступ.

Преимущества данного подхода:

1. Высокая скорость и эффективность передачи – обеспечивает малый размер передаваемых данных, который иногда даже будет помещаться в один TCP-пакет – здесь, конечно, же все зависит от вашей бизнес-логики.

2. В отличие от http, веб-сокеты не имеют ограничений на время жизни в неактивном состоянии. Это значит, что больше не надо периодически рефрешить соединение, так как его не вправе «прихлопывать» всякие прокси. Значит, соединение может висеть в неактивном виде и не требовать ресурсов. Конечно, можно возразить, что на сервере будут забиваться TCP-сокеты. Для этого достаточно использовать хороший мультиплексор, и нормальный сервер легко потянет до миллиона открытых коннектов.

3. Как известно в HTTP предусмотрено ограничение на число одновременных открытых сессий к одному серверу. Из-за этого если у вас много различных асинхронных блоков на странице, то вам приходилось делать не только серверный, но и клиентский мультиплексор. Однако это ограничение не распространяется на веб-сокеты. Открываете столько, сколько вам нужно. А сколько использовать – одно (и через него все мультиплексировать) или же наоборот – на каждый блок свое соединение – решать вам. Исходите из удобства разработки, нагрузки на сервер и клиент.

4. Возможность разрабатывать кросс-доменные приложения. AJAX имеет ряд ограничений на создание кросс-доменных запросов. WebSockets не имеет таких ограничений.

Socket.io подходит для использования на клиенте и сервере (разработанным с использованием технологии node.js). Для работы с технологией WebSockets на сервере существует множество реализаций (Jetty, EventMachine, Tornado, Shirasu, libwebsockets, SuperWebSocket).

Работа с библиотекой не представляет большой сложности.

Пример создания сервера на node.js:

Листинг 1 – Пример создание Socket.io сервера

```
var io = require('socket.io');
var http = require('http');
var app = http.createServer();
var io = io.listen(app);
app.listen(80);

io.sockets.on('connection', function (socket) {
  socket.on('eventServer', function (data) {
    console.log(data);
    socket.emit('eventClient', { data: 'Hello Client' });
  });
  socket.on('disconnect', function () {
    console.log('user disconnected');
  });
});
```

Схема работы с Socket.io на клиенте не отличается от работы с обычной библиотекой.

Пример создания клиента:

Листинг 2 – Пример создания Socket.io клиента

```
<html>
<head>
<title>Test socket.io</title>
<script src="https://cdn.socket.io/socket.io-1.2.1.js"></script>
</head>
<body>
<script>
  var socket = io.connect('http://localhost');
  socket.on('eventClient', function (data) {
    console.log(data);
  });
  socket.emit('eventServer', { data: 'Hello Server' });
</script>
</body>
</html>
```

Протоколом проводной связи (установления соединения и обмена данными между клиентом и сервером) для технологии WebSocket является RFC6455. Последние версии Chrome и Chrome для Android полностью соответствуют спецификации RFC6455, в том числе и в части отправки двоичных сообщений. Этот протокол также поддерживают браузеры Firefox 11 и Internet Explorer 10. На текущий момент технологию WebSockets поддерживают 93% браузеров [4].

Согласно последней версии спецификации, объект WebSocket также способен принимать двоичные сообщения. Форматом по умолчанию является blob.

Socket.io имеет множество методов рассылки сообщений.

Рассмотрим некоторые из них:

Листинг 3 – Пример нескольких способов рассылки сообщений

```
// отправить текущему сокету сформированному запросу (туда откуда пришла)
socket.emit('eventClient', "this is a test");

// отправить всем пользователям, включая отправителя
io.sockets.emit('eventClient', "this is a test");
```

```
// отправить всем, кроме отправителя
socket.broadcast.emit('eventClient', "this is a test");

// отправить всем клиентам в комнате (канале) 'game', кроме отправителя
socket.broadcast.to('game').emit('eventClient', 'nice game');

// отправить всем клиентам в комнате (канале) 'game', включая отправителя
io.sockets.in('game').emit('eventClient', 'cool game');

// отправить конкретному сокету, по socketid
io.sockets.socket(socketid).emit('eventClient', 'for your eyes only').
```

Использование протокола WebSocket меняет подход к применению серверных приложений. Хотя традиционные наборы серверного программного обеспечения, например LAMP, разработаны для обычного цикла запросов и ответов на базе HTTP, они часто плохо работают в условиях большого числа WebSocket-подключений.

Большое количество одновременных соединений требует принципиально новой архитектуры, которая сочетает низкую ресурсоемкость с возможностями параллельной работы. Такие архитектуры обычно создаются на основе потоковой или так называемой неблокирующей системы обмена данными.

Заключение

Веб-сокеты (Web Sockets) – это передовая технология, которая позволяет создавать интерактивное соединение между клиентом (браузером) и сервером для обмена сообщениями в режиме реального времени. Веб-сокеты, в отличие от HTTP, позволяют работать с двунаправленным потоком данных, что делает эту технологию совершенно уникальной.

ЛИТЕРАТУРА

1. WebSocket [Электронный ресурс] // Wikipedia. – Режим доступа: <https://ru.wikipedia.org/wiki/WebSocket>. – Дата доступа: 26.09.17.
2. Асинхронный веб, или Что такое веб-сокеты [Электронный ресурс] / Tproger. – Режим доступа: <https://tproger.ru/translations/what-are-web-sockets/>. – Дата доступа: 26.09.17.
3. Использование WebSocket [Электронный ресурс] / learn.javascript. – Режим доступа: <https://learn.javascript.ru/websockets>. – Дата доступа: 26.09.17.
4. WebSockets [Электронный ресурс] / caniuse. – Режим доступа: <http://caniuse.com/#feat=websockets>. – Дата доступа: 26.09.17.

УДК 004.582

WEBASSEMBLY – ПУТЬ К НОВЫМ ГОРИЗОНТАМ ПРОИЗВОДИТЕЛЬНОСТИ В WEB**М.А. БАЛАБАШ****(Представлено: Д.В. ПЯТКИН)**

Рассматривается технология WebAssembly – эффективный низкоуровневый байт-код, предназначенный для исполнения в браузере. Для компиляции кода в формат WebAssembly используется Emscripten, который позволяет достичь высокой производительности.

Технология WebAssembly – это инициатива, направленная на создание безопасного, переносимого и быстрого для загрузки и исполнения формата кода, подходящего для Web. WebAssembly – цель компиляции, у которой имеются спецификации текстового и бинарного форматов. Это означает, что другие низкоуровневые языки, такие как C/C++, Rust, Swift, и так далее, можно скомпилировать в WebAssembly. WebAssembly дает доступ к тем же API, что и браузерный JavaScript, органично встраивается в существующий стек технологий. Это отличает wasm от чего-то вроде Java-апплетов. Архитектура WebAssembly – это результат коллективной работы сообщества, в котором имеются представители разработчиков всех ведущих веб-браузеров. Для компиляции кода в формат WebAssembly используется Emscripten.

Основная часть. Emscripten – это компилятор из байт-кода LLVM в JavaScript. То есть с его помощью можно скомпилировать в JavaScript программы, написанные на C/C++ или на любых других языках, код на которых можно преобразовать в формат LLVM. Emscripten предоставляет набор API для портирования кода в формат, подходящий для веб. Этому проекту уже много лет, в основном его используют для преобразования игр в их браузерные варианты. Emscripten позволяет достичь высокой производительности благодаря тому, что он генерирует код, соответствующий стандартам Asm.js, о котором ниже, но недавно его успешно оснастили поддержкой WebAssembly.

Asm.js – это низкоуровневое оптимизированное подмножество JavaScript, обеспечивающее линейный доступ к памяти с помощью типизированных массивов и поддерживающее аннотации с информацией о типах данных. Asm.js позволяет повысить производительность решений. Это – тоже не новый язык программирования, поэтому, если браузер его не поддерживает, Asm.js-код будет выполняться как обычный JavaScript, то есть, от его использования не удастся получить прироста производительности.

WebAssembly, по состоянию на 10.01.2017, поддерживается в Chrome Canary и Firefox. Для того чтобы wasm-код заработал, нужно активировать соответствующую возможность в настройках.

В Safari поддержка WebAssembly пока в стадии разработки. В V8 wasm включен по умолчанию. Вот интересное видео о движке V8, о текущем состоянии поддержки JavaScript и WebAssembly с Chrome Dev Summit 2016.

WebAssembly это:

- **Улучшение JavaScript:** Реализуйте все критичные вещи на wasm и импортируйте его как стандартный JavaScript модуль.
- **Новый язык программирования:** WebAssembly определяет абстрактное синтаксическое дерево (как и JavaScript) в бинарном формате. Вы можете писать код и чистить его от ошибок в текстовом формате. WebAssembly легко читаем.
- **Улучшение для браузеров:** Браузеры будут понимать бинарный формат, а это значит, что разработчики смогут компилировать бинарники, которые можно сжать гораздо больше, чем используемые сегодня текстовые файлы с JavaScript. Чем меньше файл, тем быстрее загрузка. В зависимости от возможностей оптимизации времени компиляции, код на WebAssembly может передаваться и запускаться быстрее, чем на JavaScript!
- **Целевая компиляция:** Возможность другим языкам, получить первоклассную двоичную поддержку через весь стек веб-платформы.

WebAssembly добавляет вещи, которые большинство JS разработчиков не хотят видеть в JavaScript. Сама функциональность нужна, но вот в JavaScript ей места точно нет. Тем более что мы можем получить все эти функции с помощью компиляции с других языков программирования.

Фактически, WebAssembly предоставляет нам альтернативный компилятор – созданный специально для этих целей.

Теперь нам будет гораздо легче портировать код, который сильно зависит от, например, совместно используемых цепочек памяти. Представляется, что написать компилятор для WebAssembly будет легче, чем написать компилятор для JavaScript, а все потому, что первый гарантирует лучший перенос функций языка в заданное абстрактное синтаксическое дерево.

Среди всего прочего, его можно будет использовать для простой работы с тредами и SIMD (single instruction, multiple data) – проще говоря, с одним потоком команд и несколькими потоками данных.

Вы можете поставить в очередь множество блоков данных, а затем прописать одну команду для одновременной работы с ними. Это значит, что параллельная обработка потокового видео будет обрабатываться процессором. Реализуем функцию умножения на два на языке программирования C.

Листинг 1 – Реализация функции умножения на два

```
int doubler(int x) {  
    return 2 * x;  
}
```

Займемся преобразованием программы, написанной на C, в формат wasm. Для того чтобы это сделать, решено воспользоваться возможностью создания автономных модулей WebAssembly. При таком подходе на выходе компилятора мы получаем только файл с кодом WebAssembly, без дополнительных вспомогательных .js-файлов.

Такой подход основан на концепции дополнительных модулей (side module) Emscripten. Здесь имеет смысл использовать подобные модули, так как они, в сущности, очень похожи на динамические библиотеки. Например, системные библиотеки не подключаются к ним автоматически, они представляют собой некие самодостаточные блоки кода, выдаваемого компилятором.

Листинг 2 – Преобразованием программы, написанной на C, в формат wasm.

```
emcc hello_world.c -Os -s WASM=1 -s SIDE_MODULE=1 -o hello_world.c
```

После получения бинарного файла нам нужно лишь загрузить его в браузер. Для того чтобы это сделать, API WebAssembly предоставляет объект верхнего уровня WebAssembly, который содержит методы, нужные для того, чтобы скомпилировать и создать экземпляр модуля.

Листинг 3 – универсальный загрузчик модулей

```
function loadWebAssembly(filename, imports) {  
    return fetch(filename)  
        .then(response => response.arrayBuffer())  
        .then(buffer => WebAssembly.compile(buffer))  
        .then(module => {  
            imports = imports || {};  
            imports.env = imports.env || {};  
            imports.env.memoryBase = imports.env.memoryBase || 0;  
            imports.env.tableBase = imports.env.tableBase || 0;  
            if (!imports.env.memory) {  
                imports.env.memory = new WebAssembly.Memory({ initial: 256 });  
            }  
            if (!imports.env.table) {  
                imports.env.table = new WebAssembly.Table({ initial: 0, element: 'anyfunc' });  
            }  
            return new WebAssembly.Instance(module, imports);  
        });  
}
```

Сначала мы берем содержимое файла и конвертируем его в структуру данных формата ArrayBuffer. Буфер содержит исходные двоичные данные фиксированной длины. Напрямую исполнять их мы не можем, именно поэтому на следующем шаге буфер передает методу WebAssembly.compile, который возвращает WebAssembly.Module, экземпляр которого, в итоге, можно создать с помощью WebAssembly.Instance.

Листинг 4 – Пример страницы с загрузкой и исполнением модуля

```
<html>  
<head>  
<script>  
    if (!('WebAssembly' in window)) {  
        alert('you need a browser with wasm support enabled :(');  
    }  
  
    function loadWebAssembly(filename, imports) {
```

```
return fetch(filename)
  .then(response => response.arrayBuffer())
  .then(buffer => WebAssembly.compile(buffer))
  .then(module => {
    imports = imports || {};
    imports.env = imports.env || {};
    imports.env.memoryBase = imports.env.memoryBase || 0;
    imports.env.tableBase = imports.env.tableBase || 0;
    if (!imports.env.memory) {
      imports.env.memory = new WebAssembly.Memory({ initial: 256 });
    }
    if (!imports.env.table) {
      imports.env.table = new WebAssembly.Table({ initial: 0, element: 'anyfunc' });
    }
    return new WebAssembly.Instance(module, imports);
  });
}

loadWebAssembly('hello_world.wasm')
  .then(instance => {
    var exports = instance.exports;
    var doubler = exports._doubler;

    var button = document.getElementById('run');
    button.value = 'Call a method in the WebAssembly module';
    button.addEventListener('click', function() {
      var input = 21;
      alert(input + ' doubled is ' + doubler(input));
    }, false);
  });
</script>
</head>
<body>
  <input type="button" id="run" value="(waiting for WebAssembly)"/>
</body>
</html>
```

Выполнение WebAssembly занимает меньше времени, так как можно обойтись без хитростей и приемов, которые должен использовать разработчик для повышения производительности JavaScript. Кроме того, бинарный формат WebAssembly значительно ближе к машинному коду. Также в WebAssembly не требуется применение сборщика мусора, так как применяется явное управление памятью.

Заключение

Более компактное представление WebAssembly позволяет сократить время загрузки по сравнению с загрузкой даже сжатого JavaScript. Декодирование WebAssembly занимает значительно меньше времени по сравнению с парсингом исходных текстов JavaScript. Компиляция и оптимизация выполняются быстрее, так как WebAssembly ближе к машинному коду и уже прошел стадии оптимизации на этапе компиляции разработчиком. Не требуется выполнение операции повторной оптимизации, учитывающей статистику о переменных, полученную при выполнении приложения, так как в WebAssembly изначально присутствует информация о типах, которую JavaScript вынужден вычислять на ходу в зависимости от контекста. Все это делает будущее веб-технологий более производительным и еще более функциональным.

ЛИТЕРАТУРА

1. Собираем ваш первый WebAssembly-компонент [Электронный ресурс] / Хабрахабр. – Режим доступа: <https://habrahabr.ru/company/infopulse/blog/304362/>. – Дата доступа: 27.09.17.
2. WebAssembly [Электронный ресурс] // Wikipedia. – Режим доступа: <https://ru.wikipedia.org/wiki/WebAssembly>. – Дата доступа: 27.09.2017.
3. WebAssembly [Электронный ресурс] / DeveloperMozilla. – Режим доступа: <https://developer.mozilla.org/en-US/docs/WebAssembly>. – Дата доступа: 27.09.2017.

УДК 004.02

ИСПОЛЬЗОВАНИЕ CLOUDKIT В КАЧЕСТВЕ СРЕДСТВА СИНХРОНИЗАЦИИ ДАННЫХ

С.В. РЕЗАНОВ*(Представлено: канд. техн. наук И.Б. БУРАЧЕНОК)*

Рассматривается способ синхронизации данных на различных типах устройств. Дано подробное описание проблемы синхронизации и способ ее решения. Анализируются проблемы синхронизации мобильных клиент-серверных приложений и их решения с помощью CloudKit. Описаны алгоритмы оповещения устройств об изменении данных в облачном хранилище и синхронизации их с локальными базами данных

В современном мире у каждого человека есть множество электронных устройств, которые обрабатывают информацию. Клиент-серверные приложения общаются с хранилищем данных через Интернет-соединение. Мобильные устройства, такие как смартфоны, планшеты, умные часы, не могут на постоянной основе иметь соединение с сервером. Причины этому – отсутствие сети, либо плохое соединение. Также выполнять запросы получения данных через Интернет крайне нежелательно, если на устройстве низкий заряд батареи. Если устройство не может получить данных, то приложение становится неработоспособным, что негативно воспринимается пользователями. Еще одним негативным фактором для пользователей является регистрация и аутентификация. Пользователи не хотят тратить время на ввод данных и запоминание паролей, при этом не стоит упускать из виду и их безопасность. Пользователи не готовы передавать свои данные третьим лицам. Желательно, чтобы данные хранились на устройствах пользователя или на серверах компании, которой можно доверять.

Решением изложенных выше проблем может стать модель облачной синхронизации данных: на каждом устройстве пользователя находится локальная копия базы данных, а для синхронизации данных используется облачное хранилище.

Существует множество средств для хранения данных в облаке, одно из самых надежных представила компания Apple под названием iCloud. *iCloud* – облачное хранилище и сервис по облачным вычислениям, выпущенный 21 октября 2011 года. Он дает возможность хранения разнородных данных, привычных для пользователя: документы, изображения, музыку, заметки и т.п. Сервис предоставляет метод синхронизации данных для устройств, а также копию данных с носимых устройств в iCloud и последующее их восстановление на другие устройства [1]. Множество различных сервисов построено вокруг iCloud, поэтому появление программного интерфейса приложения (API) для удобной работы с ним было необходимо. Таким API стал CloudKit. Фреймворк обеспечивает аутентификацию, работу с публичной и приватной базами данных и key-value хранилище с отображением in-memory объектов.

Рассмотрим основные преимущества, согласно которым принято решение в качестве основного средства для синхронизации данных использовать CloudKit:

1) *простота*. Для работы с CloudKit не требуется какой-либо нетривиальной установки, потому что фреймворк полностью интегрирован в среду разработки Xcode и для его изначальной работы требуется только лишь аккаунт разработчика. Всю аутентификацию CloudKit производит сам. API для разработки также является тривиальным.

2) *надежность*. Еще одно преимущество CloudKit – это надежность хранения данных, CloudKit позволяет удобно организовать хранение данных приватных данных пользователя и обеспечивает безопасность приложения на проверенных решениях от Apple вместо реализации своей системы.

3) *цена*. Для каждого разработчика в программе от Apple бесплатно дается контейнер для приватных баз данных [2].

Далее рассмотрим алгоритм синхронизации данных на различных устройствах. При первом запуске приложения на устройстве, оно создает подписку на получение сообщений об изменении данных от CloudKit [3]. В момент, когда в облачном хранилище изменяются данные другим устройством, CloudKit передает сообщение сервису Apple Push Notification service (APNS). APNS сервис делает рассылку push-уведомлений всем подписанным устройствам. После получения такого уведомления приложение делает запрос к облачному хранилищу для получения актуальных данных. Схема получения данных от сервера показана на рисунке 1. Если актуализация данных не нуждается в срочности, то данную операцию можно отложить до момента, когда устройство будет находиться в Wi-Fi сети и иметь высокую степень заряда батареи.

Для актуализации данных на устройстве получать всю базу данных слишком избыточно. Этот процесс может потребовать много трафика, заряда батареи и времени. Для оптимизации актуализации данных используется система тэгов.

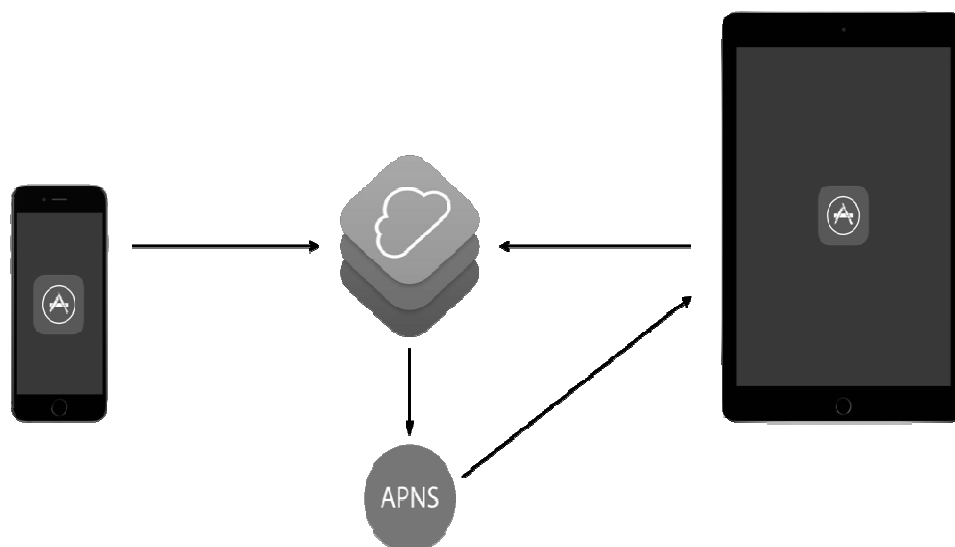


Рисунок 1. – Схема получения данных от сервера

После каждого добавления, удаления или изменения данных в CloudKit, приложение получает тэг последней операции. В момент, когда программе требуется актуализировать данные, она передает тег CloudKit и получает в ответ только те изменения, которые произошли в хранилище после получения данного тега. В конце синхронизации приложение получает новый актуальный тэг. Если при синхронизации произойдет какая-либо ошибка, то с помощью системы тегов можно будет продолжить синхронизацию данных позже, с момента наступления ошибки. Схема синхронизации данных представлена на рисунке 2.

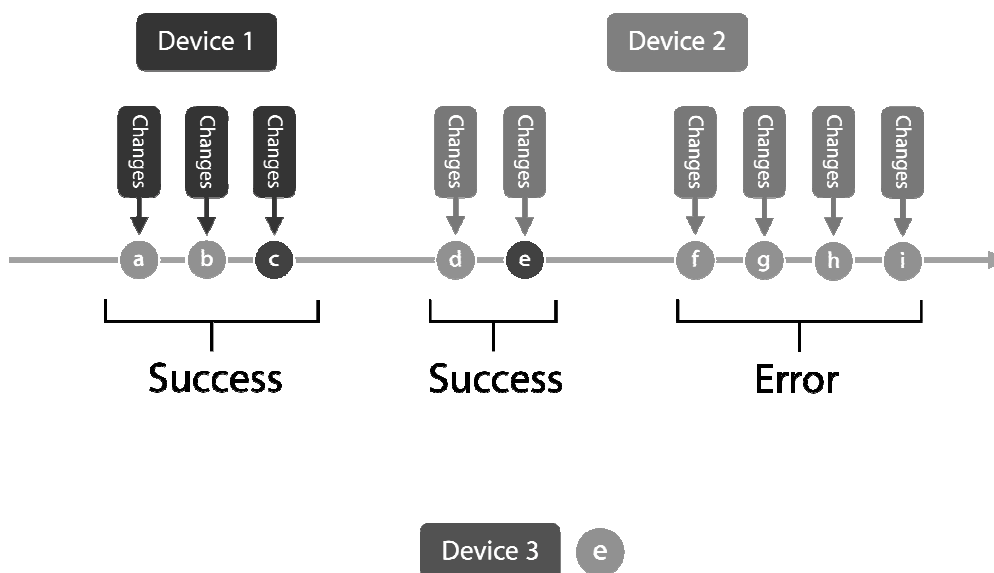


Рисунок 2. – Схема синхронизации данных

Устройство № 1 делает изменения a, b, c в облачном хранилище, а устройство № 2 – d, e, f, g, h, i. Устройство № 3 во время синхронизации получает данные от устройства № 1 и сохраняет в локальном хранилище тег "c". Затем с помощью этого тэга получает данные от второго устройства и запоминает тег "e".

Во время получения следующих изменений прервалась передача данных, и запрос выполнялся с ошибкой. После восстановления сети устройство № 3 сможет продолжить получать новые данные без избыточности.

Заключение

В результате проведенной работы исследованы проблемы синхронизации мобильных клиент-серверных приложений и их решения с помощью CloudKit. Описаны алгоритмы оповещения устройств об изменении данных в облачном хранилище и синхронизации их с локальными базами данных.

Таким образом, проведенный анализ системы облачного хранения информации CloudKit позволяет реализовать синхронизацию локальных баз данных в iOS, macOS, watchOS и tvOS приложениях.

ЛИТЕРАТУРА

1. iCloud. Forsafekeeping. And easy sharing [Electronic resource] / 2017 Apple Inc. – Mode of access: <https://www.apple.com/icloud/>. – Date of access: 24.07.2017.
2. CloudKit [Electronic resource] / 2017 Apple Inc. – Mode of access: <https://developer.apple.com/icloud/cloudkit/>. – Date of access: 24.07.2017.
3. CloudKit BestPractices [Electronic resource] / 2017 Apple Inc. – Mode of access: <https://developer.apple.com/videos/play/wwdc2016/231/>. – Date of access: 24.07.2017.

УДК 004.02

ЛОКАЛИЗАЦИЯ ПРИЛОЖЕНИЯ НА ОПЕРАЦИОННОЙ СИСТЕМЕ IOS

С.В. РЕЗАНОВ*(Представлено: канд. техн. наук И.Б. БУРАЧЕНОК)*

Рассмотрены причины необходимости локализации приложений. Подробно описаны проблемы, возникающие при локализации, и пути их решения.

Как показали исследования, русскоязычные страны замыкают пятерку стран-лидеров по количеству скачиваний мобильных приложений в AppStore и GooglePlayMarket в первом квартале 2016 года, но по выручке с продаж они не вошли и в первую десятку [1]. Это указывает на недостаточную платежеспособность населения по сравнению с другими странами. Приложения, разработанные только для русскоязычных пользователей, коммерчески не выгодны по отношению к интернациональным приложениям.

Жители Бразилии, России, Китая и Индии составляют 40% населения всего мира. Мониторинг спроса на локализацию приложений на тот или иной язык позволяет компаниям-разработчикам расширить свою клиентскую базу. На 2017 год топ 10 самых востребованных языков в Интернете выглядит следующим образом:

1. Английский
2. Китайский (упрощенный)
3. Испанский
4. Немецкий
5. Японский
6. Французский
7. Арабский
8. Португальский (бразильский)
9. Русский
10. Корейский

Рейтинг востребованности языков зависит не только от количества носителей, но и от влияния экономики страны на мировой рынок. Английский – язык международного общения и во многих странах он является официальным государственным языком [2]. Мандарин (китайский язык) является языком международного бизнеса. Это наиболее используемый язык во всем мире, а китайская экономика занимает 2-е место по номинальному значению валового внутреннего продукта (ВВП) [3]. Экономика Бразилии занимает 9-е место в мире. На французском языке в мире разговаривают около 274 млн человек, в 29-ти странах он является государственным [4]. Япония – 3-я в рейтинге стран по ВВП в мире. Она является мировым лидером в автомобилестроении и высоких технологиях, что делает знание японского языка востребованным на рынке языковых услуг. Около 240 миллионов человек разговаривает на арабском языке, и в 23 странах он является официальным [5]. Объединенные Арабские Эмираты (ОАЭ) являются мировым лидером на нефтяном рынке.

Проведенный анализ рейтинга востребованности языков позволяет сделать вывод, что коммерчески успешному приложению требуется не просто перевод, а локализация. Перевод – это интерпретация текста или речи с одного языка на другой. Она нацелена на передачу смысла информации без учета национальных особенностей, культурных ценностей и в случае с разработкой мобильных приложений – привычного опыта пользования жителей конкретной страны и особенностей целевой аудитории. Локализация включает в себя целый комплекс технических задач по адаптации кода, компонентов и добавления ресурсов. Она решает проблему различия в культуре, языке и восприятии материала разными национальностями. По сути, локализовать приложение – значит перевести его так, чтобы у пользователя не возникло никаких сложностей с пониманием и использованием продукта, с назначением его функций. То есть перевести – значит – интерпретировать, а локализовать – адаптировать.

При разработке приложения под операционную систему iOS первое, что необходимо сделать для локализации, это создать файлы Localizable.strings с переводами на поддерживаемые языки. Функция NSLocalizedString(key: String, comment: String) вернет локализованную строку согласно языку устройства. Эта функция найдет файл текущей локализации Localizable.strings и вернет строку, соответствующую параметру key [7].

Для того чтобы на всех языках текст отображался корректно, нужно использовать кодировку UTF-8 или UTF-16 и шрифты, совместимые с поддерживаемыми языками. Так как длина текста на разных языках различна, то необходимо реализовать динамическую верстку пользовательского интерфейса.

В качестве примера рассмотрим, как фраза «Сохранить как» представлена на разных языках:

ru:	Сохранить как
en:	Save as
fi:	Tallennanimellä
zh:	另存为

Числа, как и слова, тоже нуждаются в переводе. Многие разработчики забывают об этом, отображая числовые переменные в привычных для себя форматах.

Ниже приведены особенности отображения чисел для разных стран:

ru:	18 765,22
en:	18,765.22
de:	18.765,22
he:	18,765.22
el:	18.765,22
fa:	18.765.22

В качестве тысячных и дробных разделителей используются разные символы. В английском языке и иврите точка и запятая стоят совсем иначе, чем в немецком и греческом языках. А в русском языке в качестве тысячного разделителя для чисел больше 9999 используется пробел (неразрывный).

Для решения проблемы чисел компания Apple предоставила метод `localizedStringWithFormat` (format: String, arguments: CVarArg...) класса String.

В разных языках знаки валюты располагаются по-разному. Причем нормы различаются не только среди языков, но и среди языковых вариантов (в Австрии и Германии) [6]:

ru:	2,25 €
en:	€2.25
de-at:	€ 2,25
de-de:	2,25 €
lv:	€ 2,25
lt:	2,25 €

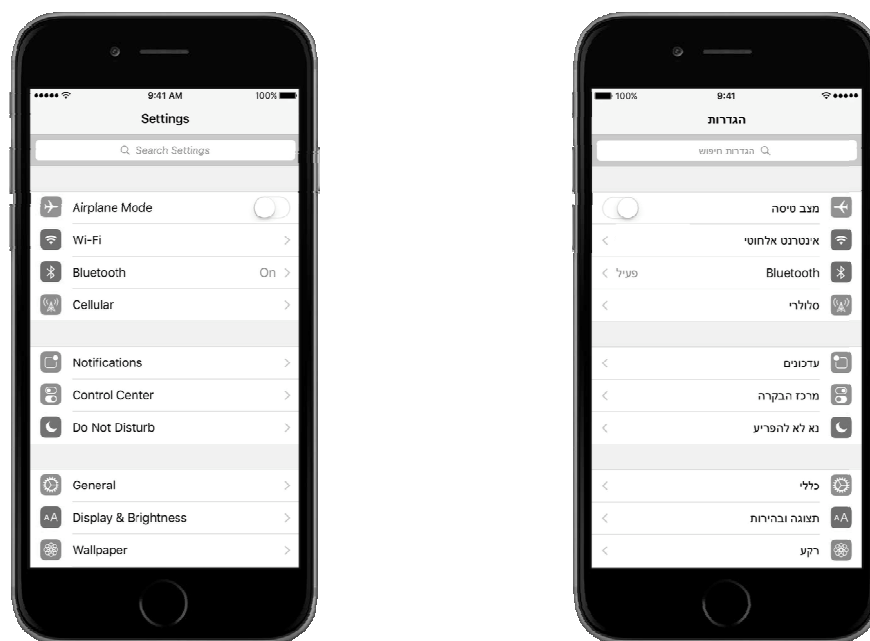
Для решения данной проблемы Apple представила класс `NumberFormatter`. В зависимости от выбранного региона в настройках iOS устройства экземпляр класса `NumberFormatter` переведет число в локализованную строку.

В разных странах используется свой формат даты и времени:

ru:	26.01.1993 9:24:00
en:	26-01-1993 9:24:00 AM
pl:	1993-01-26 9:24:00
it:	26/01/1993 09.24.00

Для форматирования дат в iOS используется экземпляр класса `DateFormatter`. Он переводит в строковый вид объект даты в зависимости от текущей локализации устройства. Также для локализации различных величин и строк используются классы: `ByteCountFormatter`, `DateComponentsFormatter`, `DateIntervalFormatter`, `PersonNameComponentsFormatter`, `MeasurementFormatter`.

Компания Apple в 2015 году представила мощный программный интерфейс приложения (API) для реализации локализации приложений для арабских стран. Жители этих стран не просто читают текст справа налево, но и воспринимают так весь контент. Для арабской локализации требуется поддерживать режим `right-to-left` (рисунок 1). Разработчикам нужно быть внимательными, так как не все элементы требуют зеркалирования [8].



Left-To-Right

Right-To-Left

Рисунок 1. – Режимы отображения интерфейса iOS приложений

Заключение

Во время повсеместного распространения мобильных устройств недостаточно иметь локализацию приложения для одной страны. Для увеличения количества пользователей, их лояльности, а также коммерческой выгоды необходимо локализовать приложение минимум для нескольких стран из списка лидеров продаж цифрового контента. Проведенные исследования возможных проблем локализации iOS приложений с детальными примерами, а также пути их решения позволят не допустить основных ошибок при локализации приложения.

ЛИТЕРАТУРА

1. РБК [Электронный ресурс] / РосБизнесКонсалтинг. – М., 2017. – Режим доступа: http://www.rbc.ru/technology_and_media/17/01/2017/587dc86a9a7947442359170e. – Дата доступа: 25.07.2017.
2. Список стран, где английский является официальным языком [Электронный ресурс] / Wikipedia2017. – Режим доступа: https://ru.wikipedia.org/wiki/Список_стран,_где_английский_является_официальным_языком. – Дата доступа: 25.07.2017.
3. Список стран по ВВП (номинал) [Электронный ресурс] / Wikipedia 2017. – Режим доступа: [https://ru.wikipedia.org/wiki/Список_стран_по_ВВП_\(номинал\)](https://ru.wikipedia.org/wiki/Список_стран_по_ВВП_(номинал)). – Дата доступа: 25.07.2017.
4. Французский язык [Электронный ресурс] / Wikipedia 2017. – Режим доступа: https://ru.wikipedia.org/wiki/Французский_язык. – Дата доступа: 25.07.2017.
5. Арабский язык [Электронный ресурс] / Wikipedia 2017. – Режим доступа: https://ru.wikipedia.org/wiki/Арабский_язык. – Дата доступа: 25.07.2017.
6. Linguistic issues concerning the euro [Electronic resource] / 2017 Wikipedia. – Mode of access: https://en.wikipedia.org/wiki/Linguistic_issues_concerning_the_euro. – Date of access: 25.07.2017.
7. Internationalization Best Practices [Electronic resource] / 2017 Apple Inc. – Mode of access: <https://developer.apple.com/videos/play/wwdc2016/201/>. – Date of access: 24.07.2017.
8. Supporting Right-to-Left Languages [Electronic resource] / 2017 Apple Inc. – Mode of access: <https://developer.apple.com/library/content/documentation/MacOSX/Conceptual/BPInternational/SupportingRight-To-LeftLanguages/SupportingRight-To-LeftLanguages.html>. – Date of access: 24.07.2017.

УДК 004.02

ВЫБОР PHP ФРЕЙМВОРКА

А.П. ГАЙДЕЛЬ*(Представлено: Д.В. ПЯТКИН)*

Рассматриваются и сравниваются три популярных PHP фреймворка: Symfony, Laravel и Yii. Если сравнивать Symfony, Laravel и Yii, все три PHP фреймворка являются отличными вариантами, которые обеспечивают full-stack разработку. Но, как представляется, Laravel является наиболее удобным фреймворком.

PHP широко используется в крупных проектах. Facebook, например, использует PHP для сохранения и создания своих внутренних систем. WordPress использует PHP, чтобы оживить свои внутренности, которые, в свою очередь, приводят в движение более чем 26% сети.

Зачем нужен PHP фреймворк?

Какой смысл использовать фреймворк вместо чистого PHP для разработки приложения? Несколько преимуществ использования фреймворков:

- PHP фреймворк делает разработку быстрее. Например, вам не нужно писать сложные запросы для извлечения данных из базы. Фреймворки включают CRUD операции (Create, Read, Update и Delete).
- Фреймворки позволяют разработчикам легко масштабировать системы.
- Код приложения краткий и с ним легко работать.
- MVC обеспечивает быструю разработку.
- Безопасность веб-приложений выше.
- Минимальное количество кода производят максимальный эффект.

Указанные преимущества являются слишком значительными, чтобы быть проигнорированными. Несмотря на это, чистый PHP может быть использован для создания любого приложения, действующие стандарты требуют разработки инструментов и навыков тайм-менеджмента для удовлетворения рыночного спроса.

Как выбрать PHP фреймворк?

Задайте себе пару вопросов, представленных ниже, они помогут вам в выборе:

1. Каковы особенности и функциональные возможности фреймворка? (Есть ли в нем то, что мне нужно?)
2. Насколько его сложно изучить?
3. Насколько высокий уровень масштабирования фреймворка?
4. Активно ли развивается и поддерживается?
5. Обеспечивается ли основа долгосрочной поддержки (LTS поддержка)?
6. Есть ли у фреймворка сильная поддержка сообщества?

Symfony, Laravel и Yii

Кратко рассмотрим претендентов:

Symfony

Symfony представляет собой набор повторно используемых компонентов PHP, что позволяет разработчикам создавать масштабируемые, высокопроизводительные приложения. Выбрать можно из 30 компонентов, разработчик имеет полную свободу экспериментировать и работать в среде RAD. Symfony API позволяют также легко интегрироваться с приложениями сторонних разработчиков, и он может быть использован с популярными фреймворками, такими как AngularJS.

Многие популярные проекты, в том числе Drupal и phpBB, также использовали Symfony. На самом деле, Laravel, самый популярный PHP фреймворк, это построен на Symfony.

Laravel

Laravel имеет отличное сообщество и опережает всех в списке наиболее популярных фреймворков. (Одним из ведущих разработчиков Laravel на Livcoding.tv является Sfiskell.)

В мае 2015 года Laravel объявила, что версия 5.1 будет включать долгосрочную поддержку в течение двух лет. Версия 5.2 была представлена в декабре 2015 года. Многие хостинг компании предоставляют поддержку Laravel и предлагают хостинг решения для приложений на Laravel.

Yii

Yii является безопасным, быстрым, высокопроизводительным фреймворком для веб-разработки и разработки приложений.

Yii использует менеджер зависимостей для PHP для обработки различных зависимостей и установок (подробнее об этом позже). Yii также является самым быстрым PHP фреймворком.

Еще одна интересная особенность Yii интегрирован с JQuery. Интеграция позволяет front-end разработчикам быстро охватить фреймворк. Подобно Symfony, Yii также использует компоненты, обеспечивающие быструю разработку приложений.

Все три фреймворка идеально подходят для создания веб-приложений 2.0, но каждый из них служит разным целям.

Рассмотрим их особенности.

Поддержка баз данных

Symfony 2 обеспечивает лучшую поддержку баз данных. Вы можете работать с массивом баз данных, в том числе NoSQL и DynamoDB. Yii и Laravel также полезны, но они поддерживают меньше, чем базы данных Symfony. Поддержка баз данных каждым отдельным фреймворком отображена в таблице.

Framework	Laravel	Yii	Symfony 2
Database	SQLite MySQL PostgreSQL Redis Microsoft BI MongoDB	MySQL SQLite Microsoft BI Oracle PostgreSQL MongoDB	Microsoft BI MongoDB MySQL NoSQL PostgreSQL CouchDB DynamoDB GemFire GraphDB MemBase MemCacheDB Oracle Apache Jackrabbit

Сходства

Рассмотрев различия между Фреймворками, выделим их сходства.

Все три фреймворка являются full-stack PHP фреймворками и предлагают функциональные возможности для создания веб-приложений, от front-end до back-end

- Проекты с открытым исходным кодом и их исходный код можно найти на GitHub
- Фреймворки хорошо документированы и поддерживаются большими сообществами
- Каждый из них поддерживает ORM (Object Relationship Mapping)
- Они являются надежными и безопасными для создания веб-приложений 2.0

Если сравнивать Symfony, Laravel и Yii, все три PHP фреймворка являются отличными вариантами, которые обеспечивают full-stack разработку. Но, по нашему мнению, Laravel является наиболее удобным фреймворком.

Тем не менее Symfony и Yii тоже отличные варианты. Symfony хорошо разработан и имеет более крупное, зрелое сообщество. Yii является надежным, безопасным и хорошо выполняет свою работу.

ЛИТЕРАТУРА

1. Фреймворк [Электронный ресурс] / Википедия – Свободная энциклопедия. Режим доступа: <https://ru.wikipedia.org/wiki/Фреймворк>. – Дата доступа: 24.09.2017.
2. 10 лучших PHP-фреймворков для веб-проектов [Электронный ресурс] / kultprosvet. – Режим доступа: <https://kultprosvet.net/ru/blog/10-luchshih-php-freymvorkov-dlya-veb-proektov>. – Дата доступа: 24.09.2017.
3. Laravel vs Symfony – 5 Шагов к Выбору PHP Фреймворка [Электронный ресурс] / devsize.ru. – Режим доступа: <https://devsize.ru/laravel-vs-symfony-5-steps-choosing-framework>. – Дата доступа: 25.09.2017.

УДК 004.02

СПОСОБЫ ОПТИМИЗАЦИИ БАЗ ДАННЫХ ПРИ РАБОТЕ С MYSQL

А.П. ГАЙДЕЛЬ

(Представлено: Д.В. ПЯТКИН)

Приведены практические советы использования MySQL и проектирования баз данных. Проблемы при использовании MySQL можно разделить на три группы: использование индексов, структура базы данных, SQL запросы.

Работа с базой данных занимает большую часть работы практически любого сайта. И именно работа с БД чаще всего является узким местом веб-приложений.

Проблемы при использовании MySQL

Проблемы при использовании MySQL можно разделить на три группы (в порядке значимости):

1. Неиспользование или неправильное использование индексов.
2. Неправильная структура БД.
3. Неправильные \ неоптимальные SQL запросы.

Остановимся на каждой из этих групп подробнее.

Использование индексов

Неиспользование или неправильное использование индексов – это то, что чаще всего замедляет запросы.

Советы по использованию индексов:

Не нужно индексировать все подряд. Довольно часто, не понимая смысла, многие просто индексируют все поля таблицы. Индексы ускоряют выборки, но замедляют вставки и обновления строк, поэтому выбор каждого индекса должен быть осмыслен.

- Один из основных параметров, характеризующий индекс — селективность(selectivity) — количество разных элементов в индексе. Нет смысла индексировать поле, в котором два-три возможных значения. Пользы от такого индекса будет мало.
- Выбор индексов должен начинаться с анализа всех запросов к данной таблице. Очень часто после такого анализа вместо трех-четырех индексов можно сделать один составной.
- При использовании составных индексов порядок полей в индексе имеет определяющее значение.
- Не забывайте про покрывающие(covering) индексы. Если все данные в запросе могут быть получены из индекса, то MySQL не будет обращаться непосредственно к таблице. Подобные запросы будут выполняться очень быстро. Например, для запроса `SELECT name FROM user WHERE login='test'` при наличии индекса (login, name) обращения к таблице не потребуются. Порой имеет смысл добавить в составной индекс дополнительное поле, которое сделает индекс покрывающим и ускорит запросы.
- Для индексов по строкам часто достаточно индексировать лишь часть строки. Это может значительно уменьшить размер индекса.
- Если % стоит в начале `LIKE(SELECT * FROM table WHERE field LIKE '%test')` индексы использоваться не будут.
- `FULLTEXT` индекс используется только с синтаксисом `MATCH ... AGAINST`.

Структура БД

Грамотно спроектированная БД – залог быстрой и эффективной работы с базой. С другой стороны, плохо продуманная БД – это всегда проблема для разработчиков.

Советы по проектированию БД:

Используйте минимально возможные типы данных. Чем больше тип данных, тем больше таблица, тем больше обращений к дискам нужно для получения данных. Используйте удобную процедуру: `SELECT * FROM table_name PROCEDURE ANALYSE()`; для определения минимально возможных типов данных.

- На этапе проектирования соблюдайте нормальные формы. Часто программисты прибегают к денормализации уже на этом этапе. Однако в большинстве случаев в начале проекта не ясно, чем это может закончиться. Денормализовать таблицу гораздо проще, чем страдать от неоптимально денормализованной. Да и JOIN порой работает быстрее, чем неверно денормализованные таблицы.

- Не используйте NULL столбцы кроме случаев, когда они вам осознанно нужны.

SQL запросы

Избегайте запросов в цикле. SQL – язык множеств, и к написанию запросов нужно подходить не языком функций, а языком множеств.

- Избегайте * (звездочки) в запросах. Перечислите именно те поля, которые вы выбираете. Это сократит количество выбираемых и пересылаемых данных. Кроме этого, не забывайте о покрывающих индексах. Даже если вы действительно выбираете все поля в таблице, лучше их перечислить. Во-первых, это повышает читабельность кода. При использовании звездочек невозможно узнать, какие поля есть в таблице, без заглядывания в нее. Во-вторых, сегодня в вашей таблице пять INT столбцов, а через месяц добавилось еще одно TEXT и BLOB, а звездочка как была, так и осталась.

- При страничном выборе для получения общего количества записей используйте SQL_CALC_FOUND_ROWS и SELECT FOUND_ROWS(); При использовании SQL_CALC_FOUND_ROWS MySQL кеширует выбранное количество строк (до применения LIMIT) и при SELECT FOUND_ROWS() только отдает это закешированное значение без необходимости повторного выполнения запроса.

- Не забывайте, что у INSERT есть синтаксис для множественной вставки. Один запрос будет выполняться на порядок быстрее, чем множество запросов в цикле.

- Используйте LIMIT там, где вам не нужны все данные.

- Используйте INSERT... ON DUPLICATE KEY UPDATE... вместо выборки и INSERT или UPDATE после нее, а также часто вместо REPLACE.

- Не забывайте про замечательную функцию GROUP_CONCAT. Она может выручить при сложных запросах.

Ведется много споров на счет эффективности и надежности использования некоторых из описанных методов, но окончательное решение за вами. В любом случае, теперь есть варианты, над которыми стоит задуматься.

ЛИТЕРАТУРА

1. Оптимизация запросов СУБД [Электронный ресурс] / Википедия – Свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Оптимизация_запросов_СУБД. – Дата доступа: 22.09.2017.
2. Оптимизация структуры базы данных [Электронный ресурс] / mysql.ru. – Режим доступа: <http://www.mysql.ru/docs/mysql-man-4.0-ru/mysql-optimisation.html>. – Дата доступа: 22.09.2017.
3. Оптимизация в MySQL [Электронный ресурс] / OpenNet. – Режим доступа: <https://www.opennet.ru/docs/RUS/mysqlcli/glava14.html>. – Дата доступа: 23.09.2017.

УДК 621.37:681.586

УСТРОЙСТВО УПРАВЛЕНИЯ ТЕПЛИЦЕЙ НА ОСНОВЕ МИКРОКОНТРОЛЛЕРА

С.Ю. ЗМИТРОВИЧ

(Представлено: канд. физ.-мат. наук, доц. С.А. ВАБИЩЕВИЧ)

Рассматривается встраиваемая микроконтроллерная система управления теплицей. Управление теплицей осуществляется путем анализа поступающих от датчиков сигналов на микроконтроллер и принятия определенных решений, которые выбирает оператор. Система управляет нагрузками, которые могут быть представлены приводами, насосами, светильниками, дверьми.

С давних пор важную роль в жизнедеятельности человека занимает сельское хозяйство. Именно оно обеспечивает человечество сырьем для пищевой промышленности. И одним из важных изобретений в этой области было изобретение теплицы, которая помогает аграриям добиваться определенных параметров климата искусственной среды в любую пору года, в которой, в свою очередь, и проходит полный цикл выращивания растений. Однако за таким сооружением необходимо следить, так как если не соблюдать параметры климата, растения или их плоды могут погибнуть. Для этого необходим обученный персонал или автоматизированная система, которая сама будет принимать решения исходя из полученных от датчиков данных. Автоматизированные системы позволяют повысить производительность труда, облегчить процессы управления, ограничить возможность человеческой ошибки [1].

Целью данной работы состоит в проектировании микроконтроллерного устройства для управления теплицей. На рисунке 1 представлена структурная схема устройства.

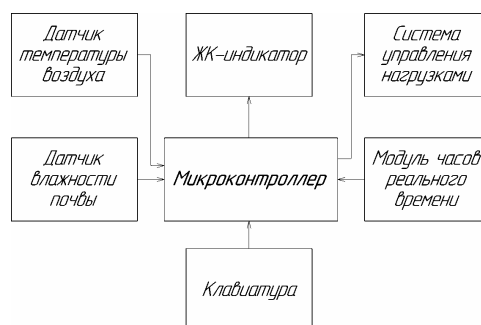


Рисунок 1. – Структурная схема устройства управления теплицей

Основа системы управления – микроконтроллер, который берет на себя все основные задачи по управлению, занимается обменом данными с датчиковой аппаратурой и модулем часов реального времени, полученную информацию о параметрах сравнивает с ранее записанной в его память, при каких-либо отклонениях воздействует управляющими командами на систему управления нагрузками, которая, в свою очередь, может управлять электродвигателями, насосами, светильниками т.д. Для отображения текстовой информации используется ЖК-индикатор, он необходим для упрощения взаимодействия оператора с микроконтроллером через клавиатуру.

Принцип работы. Принципиальные схемы системы управления для теплицы и системы управления нагрузками представлены на рисунках 2 и 3. Микроконтроллер (DD2) (рис. 2) системы управления теплицей связан с датчиками температуры (DA1, DA3, DA5), влажности (DA2, DA4) и клавиатурой (кнопки S1-S12 и делители напряжения R1-R6, R8-R11, R13-R16, R19) по однопроводным линиям, которые подсоединены к аналого-цифровому преобразователю через соответствующие, настроенные на аналоговый режим работы, порты А и Е [2–3].

Модуль часов реального времени (DD1) (рис. 3) подключается к микроконтроллеру через I2C шину передачи данных. Для обеспечения логической единицы, во избежание наводок и помех на шине, ее подтягивают к питанию (+5В) через резисторы (R17-R19), чтобы ограничить втекающие токи [4].

ЖК-индикатор (HG1) подключается к микроконтроллеру через цифровые порты ввода-вывода, через которые осуществляется управление и передача текстовой информации [5].

По питанию микроконтроллера стоят сглаживающие конденсаторы, которые необходимы для уменьшения пульсаций напряжения и электромагнитных помех.

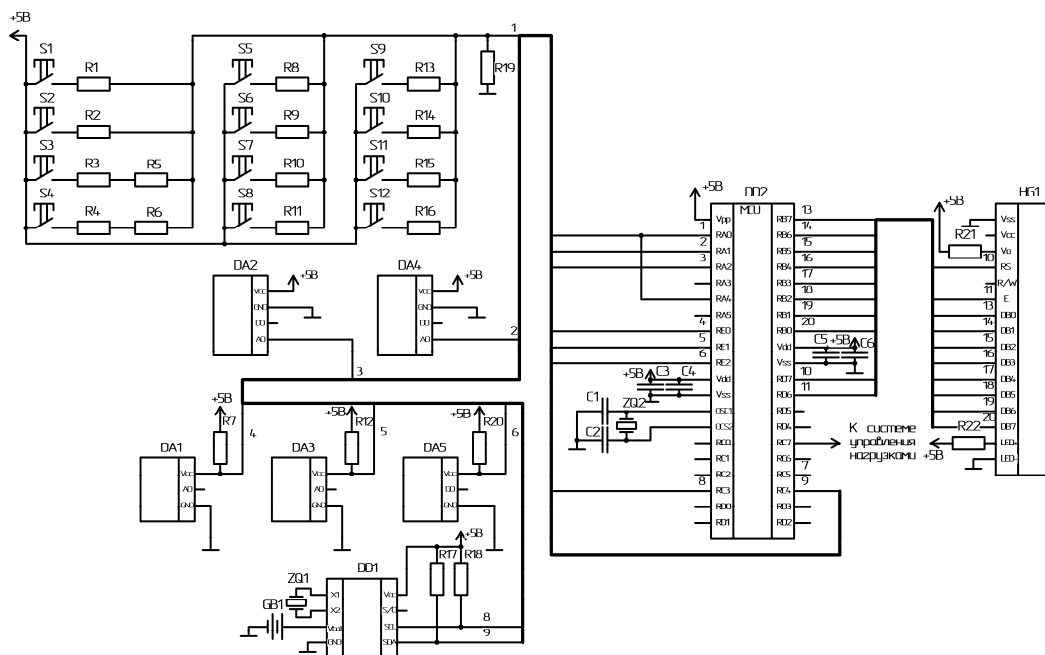


Рисунок 2. – Принципиальная схема системы управления теплицей

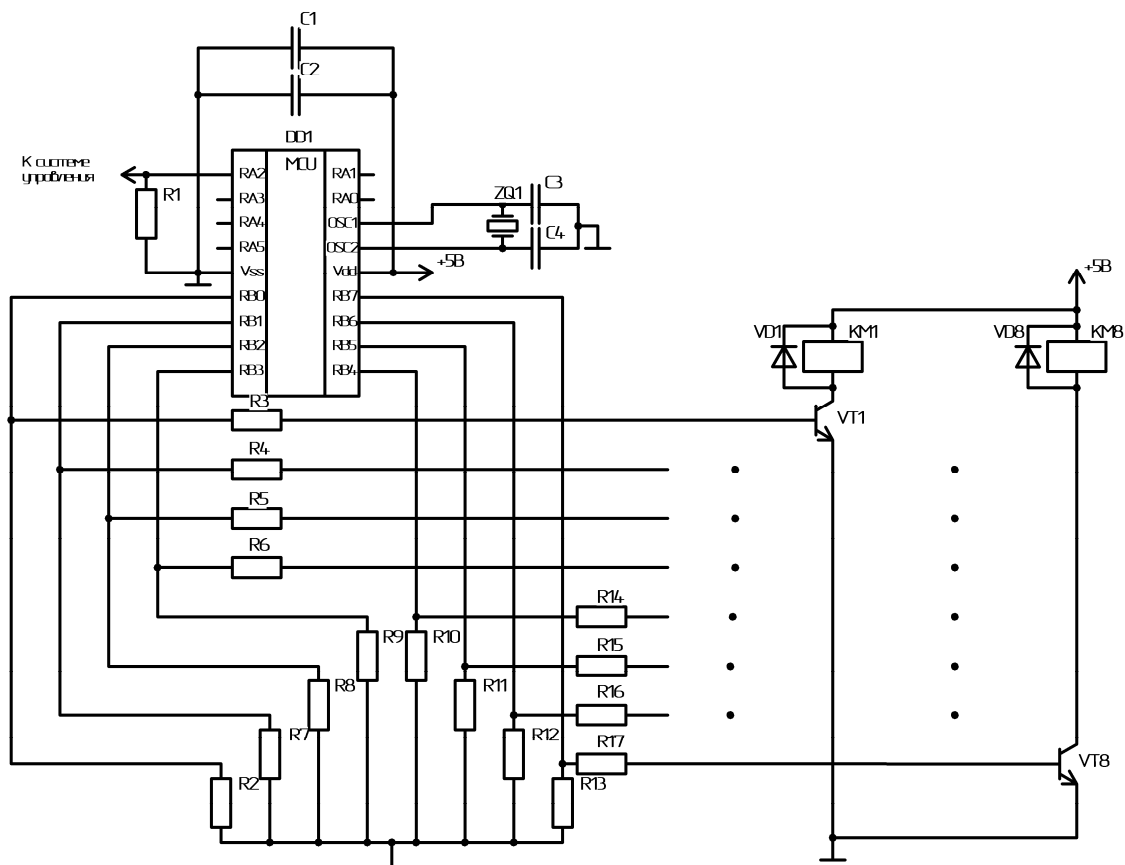


Рисунок 3. – Принципиальная схема системы управления нагрузками

Микроконтроллер системы управления нагрузками (DD1) управляет включением/выключением нагрузок через цифровые порты ввода/вывода, подавая сигналы на базы транзисторов (VT1-VT8) через резисторы (R3-R6, R14-R17). При подаче определенного сигнала на базу транзистора он либо откроется и пропустит через себя ток, проходящий через электромагнитные реле (KM1-KM8), которые, в свою оче-

редь, замыкают свои контакты и включают подключенные к ним нагрузки, либо закроеся [6]. При отключении электромагнитных реле в них образуются ЭДС самоиндукции, направление которой противоположно направлению протекающего до этого тока. ЭДС самоиндукции вредно для транзисторов, и оно может разрушить их переходы. Для того чтобы этого избежать используются шунтирующие диоды (VD1-VD8). Для обеспечения логического нуля, во избежание появления наводок или помех, используются подтягивающие резисторы (R2, R7-R13) [7].

Заключение

Рассмотрены структурная и принципиальная схемы системы управления микроклиматом для теплицы, принцип их работы. Микроконтроллерная система отслеживает параметры температуры, влажности и времени, управляет системой управления нагрузками. Есть возможность управления системой оператором, взаимодействуя через клавиатуру и ЖК-индикатор, записи в нее необходимых для проверки вышеперечисленных параметров и включения нагрузок. К системе управления нагрузками возможно подключать электродвигатели, насосы, светильники и т.д.

Система промоделирована в среде ProteusProfessional, которая показала свою полную работоспособность.

ЛИТЕРАТУРА

1. Уилмсхерст, Т. Разработка встроенных систем с помощью микроконтроллеров PIC / Т. Уилмсхерст, Ю.А. Шпак. – Киев : Мк-Пресс, 2015. – 543 с.
2. Electronic Components Datasheet Search [Электронныйресурс] / DS1307 Datasheet. – Режим доступа: <http://datasheets.maximintegrated.com/en/ds/DS1307.pdf>.
3. Electronic Components Datasheet Search [Электронныйресурс] / PIC16F877A Datasheet. – Режим доступа: <http://www.alldatasheet.com/datasheet-pdf/pdf/82338/MICROCHIP/PIC16F877A.html>.
4. Electronic Components Datasheet Search [Электронныйресурс] / HD44780 Datasheet. – Режим доступа: <http://www.alldatasheet.com/datasheet-pdf/pdf/HITACHI/HD44780.html>.
5. Electronic Components Datasheet Search [Электронныйресурс] / LM335Z Datasheet. – Режим доступа: <http://www.alldatasheet.com/datasheet-pdf/pdf/8635/NSC/LM335Z.html>.
6. Electronic Components Datasheet Search [Электронныйресурс] / PIC16F628A Datasheet. – Режимдоступа: <http://www.alldatasheet.com/datasheet-pdf/pdf/74968/MICROCHIP/PIC16F628A.html>
7. Хоровиц, П. Искусство схемотехники / У. Хилл, П. Хоровиц. – М. : Бином, 2015. – 704 с.

УДК 621.37:681.586

МИКРОКОНТРОЛЛЕРЫ ВО ВСТРОЕННЫХ СИСТЕМАХ УПРАВЛЕНИЯ**С.Ю. ЗМИТРОВИЧ***(Представлено: канд. физ.-мат. наук, доц. С.А. ВАБИЩЕВИЧ)*

Рассматривается возможность использования микроконтроллеров во встроенных системах управления, их архитектура, технические характеристики, периферия. На примере PIC16F877A приводятся примеры создания подпрограмм, с помощью которых возможно реализовать управление микроконтроллером.

Микроконтроллеры – это производные от микропроцессоров интегральные микросхемы с ЦПУ, памятью и устройством ввода/вывода. Есть следующие микроконтроллеры: 8-битные: PIC12F675, PIC16F877A, ATtiny2313 и др.; 16-битные: PIC24FJ64GA006, ATmega384A1-AU, dsPIC33FJ12GP201; 32-битные: SAM7X512, STM32, PIC32MX340F128H. С их изобретением появилась возможность проектирования сложных управляющих систем относительно небольших и малых размеров, которые помогают человеку облегчить его быт. Работают данные интегральные микросхемы точно по записанной в их память программе, без каких-либо отклонений от алгоритма, если не учитывать какие-либо сбои в их работе.

Встроенные системы управления – системы, у которых основной функцией является функция управления, реализованная на микроконтроллерах. В современном мире повсюду можно найти встроенные системы, они могут быть как в часах, микроволновых печах, так и в холодильниках, автомобилях. Держа в руках какие-либо предметы, человек может и не догадаться о том, что в них может присутствовать одновременно несколько встроенных систем, которые чем-либо управляют, анализируют, вычисляют.

Микроконтроллеры можно сопрягать с различными видами датчиковой аппаратуры, после чего отображать информацию, полученную от них, на дисплее, светодиодных индикаторах, ЖК-индикаторах и т.п. Также микроконтроллеры могут принимать определенные решения при анализе полученных данных, и отправлять управляющие команды на различные подвластные ей системы.

Параметры микроконтроллеров. Микроконтроллеры бывают нескольких разрядностей: 8-битные, 16-битные, 32-битные. Также каждый из них имеет архитектуру, от которой и зависят возможности микроконтроллеров. В зависимости от числа используемых инструкций микроконтроллеров выделяют две основные архитектурные группы: RISC и CISC. Различие этих групп заключается в том, что RISC – это тщательный подбор инструкций, которые можно выполнить за один такт тактового генератора, а CISC – архитектура менее быстродействующих инструкций, которые направлены на упрощение программ.

Программирование микроконтроллеров может осуществляться как на высокоуровневых (C++, C, MicroPascal, Basic и т.д.), так и на низкоуровневом языке программирования (Assembler). Assembler микроконтроллера PIC16F877A имеет в своем распоряжении 35 инструкций [1; 2].

Технические характеристики микроконтроллера PIC16F877A [3-4]:

Память программы: 8192 инструкции; SRAM: 368 байт; EEPROM: 256 байт; портов ввода-вывода: 33; аналоговых портов: 8; PWM: 2; таймеры 8/16: 2/1; 2 компаратора; USART, I2C, SPI; максимальная частота тактового генератора: 20 МГц.

Рассмотрим архитектуру микроконтроллера PIC16F877A, представленную на рисунке, и кратко опишем ее.

Центральное процессорное устройство, обведенное пунктиром на схеме, представлено арифметико-логическим устройством (АЛУ), служащим для выполнения арифметических и логических преобразований над данными; регистром состояния STATUS, в котором хранятся все данные о только что выполненной инструкции; рабочим регистром W, который используется во всех операциях АЛУ.

Порты A, B, C, D, E служат для ввода и вывода цифровой и аналоговой информации, также некоторые входы используются для программирования микроконтроллера, для использования таймеров, широтно-импульсного модулирования, компараторов и шин передачи данных.

Регистр косвенной адресации (FSR), расположенный во всех банках памяти микроконтроллера, может быть использован как регистр связи с другими регистрами, то есть в него может быть записан номер ячейки другого регистра, с которым необходимо взаимодействовать.

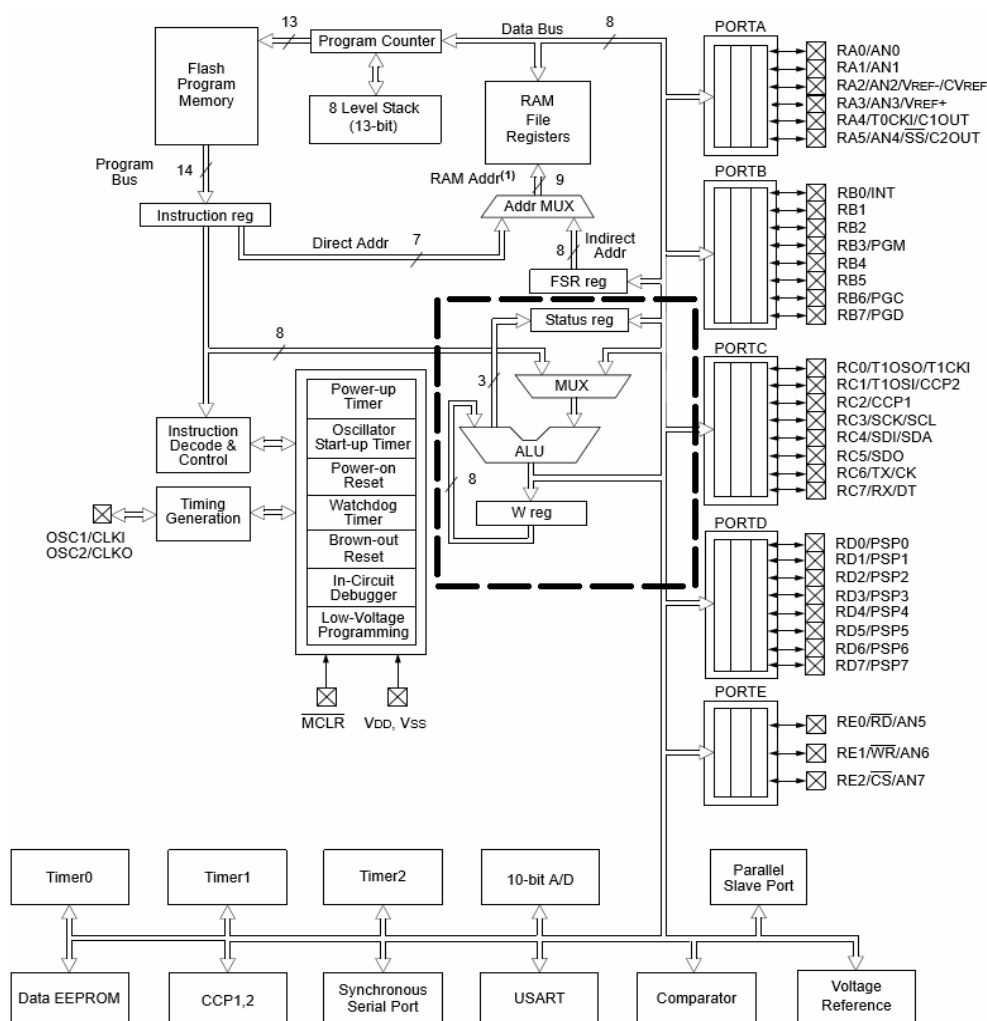
USART, SPI, I2C – шины передачи данных, которые служат для взаимодействия с другими устройствами, подключаемыми к этим шинам.

Счетчики/Таймеры могут быть применены в измерении времени или, например, в подсчете продукции проходящей по конвейеру.

EEPROM(ЭСППЗУ) – Электрически стираемое программируемое постоянное запоминающее устройство, часть памяти микроконтроллера, в которой данные могут быть записаны при написании программы микроконтроллера или при работе микроконтроллера под контролем программы.

Память программы (FlashprogramMemory) – предназначена для хранения кода программы, который можно занести через внешний программатор.

Адресный мультиплексор и мультиплексор – устройства, предназначенные для выбора регистров, с которыми будет взаимодействовать центральный процессор.



Архитектура микроконтроллера PIC16F877A

Аналого-цифровой преобразователь (А/Д) – модуль, преобразующий аналоговый тип сигнала в цифровой, является 10-разрядным. Это означает, что максимальное преобразованное число может равняться 1023.

Примеры программирования микроконтроллера

Программа инициализация портов:

...Программа...

bcf status,5 ; Переключение на 0 банк памяти

clrporta ; Очистка порта *A*, установка на всех разрядах логического нуля

clrfportb ; Очистка порта В, установка на всех разрядах логического нуля

`clrfportc` ; Очистка порта C, установка на всех разрядах логического нуля

`clrfportd` ; Очистка порта D, установка на всех разрядах логического нуля

`clrfrpote` ; Очистка порта E, установка на всех разрядах логического нуля

bsf status,5 ; Переключение на 1 банк памяти

movlwb'11111111'; Запись в рабочий регистр значения 255

movwfrisa ; Перенос значения из рабочего регистра в регистр состояния порта,
; установка всех разрядов порта A на вход

movlwb'00001111'; Запись в рабочий регистр значения 15

`movwfrise` ; Перенос значения из рабочего регистра в регистр состояния порта,

```

; установка первый четырех разрядов порта E на вход
clrfrisc ; Очистка значения в регистре состояния порта B, установка всех разрядов порта на выход
clrfrisc ; Очистка значения в регистре состояния порта C, установка всех разрядов порта на выход
movlw .0 ; Запись в рабочий регистр значения 255
movwftrisd ; Перенос значения из рабочего регистра в регистр состояния порта,
; установка всех разрядов порта D на выход

```

...Программа...

End.

У начинающих программистов возникают трудности при программировании на ассемблере, так как появляются ошибки при компилировании из-за того, что не был переключен лист памяти программы.

Подпрограмма решения этой проблемы:

...Программа...

...Память страницы заполнена, необходимо переключение...

```

bsf PCLATH,3 ; Установка 3 бита состояние логической 1
; Переход на 1 страницу памяти [0800h-0FFFh]

```

```

goto perehod1 ; Переход на метку «perehod1»

```

...Программа...

...Память страницы заполнена, необходимо переключение...

```

bsf PCLATH,3 ; Установка 3 и 4 битов регистра в состояние логической 1

```

```

bsf PCLATH,4 ; Переход на 3 страницу памяти [1800h-1FFFh]

```

```

goto perehod3 ; Переход на метку «perehod3»

```

...Программа...

End.

Подпрограмма проверки нажатия клавиши подключенной к 3 разряду порта A:

...Программа...

btfsporta, 3 ; Проверка равенства 3 разряда порта A логической единице, если равно, то выполняется следующая инструкция, если нет, то следующая инструкция пропускается

```

goto metka ; Переход на метку «metka»

```

```

$-2 ; Переход на две строки вверх по коду

```

...Программа...

End.

Подпрограмма взаимодействия с регистром косвенной адресации (FSR):

...Программа...

```

movlw ReG ; Запись в рабочий регистр адреса регистра «ReG»

```

```

movwf FSR ; Перенос значения из рабочего регистра в регистр FSR

```

```

movlw 10h ; Запись значения 10h в рабочий регистр

```

```

metka decfsz INDF,1 ; Декрементирует значение регистра адрес которого записан в FSR

```

goto met ; Если значение регистра «ReG» при выполнении предыдущей инструкции не равняется 0, то выполняется переход на метку «met»

```

goto phd ; Если же значение регистра «ReG» равняется 1, то выполняется переход на метку «phd»

```

...Программа...

Заключение

В результате проведенного исследования рассмотрены основные технические характеристики, архитектура и ее основные элементы микроконтроллера PIC16F877A, который можно использовать при проектировании встроенных систем. Приведены примеры создания подпрограмм управления и адресации. Данный микроконтроллер апробирован в системе управления теплицей.

ЛИТЕРАТУРА

1. Катцен, С. PIC-микроконтроллеры. Полное руководство / С. Катцен, А.В. Евстифеева. – М.: ДМК Пресс, 2014. – 651 с.
2. Уилмсхерст, Т. Разработка встроенных систем с помощью микроконтроллеров PIC / Т. Уилмсхерст, Ю.А. Шпак. – Киев : Мк-Пресс, 2015. – 543 с.
3. Хоровиц, П. Искусство схемотехники / У. Хилл, П. Хоровиц. – М. : Бином, 2015. – 704 с.
4. Electronic Components Datasheet Search [Электронный ресурс] / PIC16F877A Datasheet. – Режим доступа: <http://www.alldatasheet.com/datasheet-pdf/pdf/82338/MICROCHIP/PIC16F877A.html>

УДК 621.396

ВИДЫ РАССЕЯНИЯ СИГНАЛОВ В ОПТИЧЕСКИХ КАБЕЛЯХ

Е.В. СМЕТАНА

(Представлено: канд. техн. наук, доц. В.Ф. ЯНУШКЕВИЧ)

Проведен анализ видов рассеяния сигналов в оптических кабелях. Рассмотрен принцип измерения характеристик оптических кабелей на основе определения интенсивности и времени задержки импульсных сигналов.

В современном мире важную роль играют линии связи, физическая среда передачи которых может представлять собой кабель (набор проводов, изолированных и защищенных оболочкой). Кроме кабеля физической средой передачи данных может быть земная атмосфера или космическое пространство, через которые распространяются электромагнитные волны. В зависимости от среды передачи данных кабели связи можно разделить на три группы:

1) *проводные (воздушные) кабели связи* – это провода без изолирующих и экранирующих оплеток, проложенные между столбами и висящие в воздухе. Они проводят телефонные и телеграфные сигналы. Скоростные свойства и помехозащищенность низкие;

2) *радиоканалы земных и спутниковых кабелей связи* – образуются с помощью передатчика и приемника радиоволн. Могут использоваться для организации сетей в пределах больших помещений типа ангаров или павильонов – там, где использование обычных кабелей связи затруднено или нецелесообразно;

3) *кабельные* – состоят из проводников, заключенных в несколько слоев изоляции. В компьютерных сетях используют три основных типа кабеля:

- витая пара (скрученные пары медных проводников);
- коаксиальный кабель;
- оптические.

Оптические кабели (ОК) – это вид устройств передачи информации данных, при котором информация передается по оптическим диэлектрическим световодам. В настоящее время ОК считаются самой совершенной физической средой для передачи информации, а также самой перспективной средой для передачи больших потоков информации на значительные расстояния [1].

Преимущества оптических кабелей связи (ОКС), обуславливающие выбор именно ОКС как основу построения сети:

- *широкополосность*, обусловленная чрезвычайно высокой несущей частотой ($F_0 = 10^{14}$ Гц). По ОКС можно передавать информацию со скоростью порядка 1,2 млрд. бит данных в секунду;
- *очень малое затухание*. (0,2–0,3 дБ на длине волны 1,55 мкм на 1 км) светового сигнала в кабеле позволяет строить ОК длиной до 100 км и более без ретрансляции сигналов;
- *устойчивость к электромагнитным помехам* со стороны окружающих медных кабельных систем, электрического оборудования (линии электропередачи, электродвигательные установки и т.д.) и погодных условий;
- *защита от несанкционированного доступа*. Информацию, передающуюся по оптическим кабелям связи, практически нельзя перехватить неразрушающим способом;
- *электробезопасность*. Из-за отсутствия искрообразования оптический кабель повышает взрыво- и пожаробезопасность сети, что особенно актуально на химических, нефтеперерабатывающих предприятиях, при обслуживании технологических процессов повышенного риска;
- *малый вес и объем*. Оптические кабели имеют меньший вес и объем по сравнению с медными кабелями в расчете на одну и ту же пропускную способность. Например, 900-парный телефонный кабель диаметром 7,5 см может быть заменен одним кабелем с диаметром 0,1 см. Если кабель «одеть» во множество защитных оболочек и покрыть стальной ленточной броней, его диаметр будет 1,5 см, что в несколько раз меньше рассматриваемого телефонного кабеля;
- *долговечность*. Срок службы ОКС составляет не менее 25 лет.

Недостатки оптических кабелей связи (ОКС):

- *относительно высокая стоимость сварки оптических кабелей*. Для этого требуется прецизионное, а следовательно, дорогое, технологическое оборудование. Как следствие, при обрыве оптического кабеля затраты на восстановление ОКС выше, чем при работе с медными кабелями;
- *относительно высокая стоимость активных элементов ОКС*, преобразующих электрические сигналы в свет и свет в электрические сигналы [2].

При монтаже и обслуживании оптических кабелей невозможно обойтись без проведения ряда измерений. Конкретный набор параметров зависит от выполняемых работ. Самым типичным для этапа монтажа является измерение затухания как всей линии, так и отдельных сростков, выполненных с помощью сварки. На этапе пусконаладочных работ и эксплуатации определяются уровни мощности оптического излучения на выходе передатчика и входе приемника, а также фиксируется коэффициент ошибок. При проведении кроссовых работ встает задача идентификации кабелей и их окончаний, проверки исправности коммутационных шнуров и правильности кроссировки (просветка, аналог «прозвонки» на металлических кабелях). В случае обнаружения каких-либо проблем производится диагностика кабелей с помощью оптического рефлектометра [2].

Виды рассеяния сигналов в оптических кабелях. В современном мире оптические кабели связи получают все более широкое распространение, однако для корректной работы необходимо точно отслеживать их параметры. Для этой цели служат оптические рефлектометры [3].

Принцип работы прибора основан на анализе отраженных оптических импульсов, излучаемых рефлектометром в оптический кабель. Измерения с помощью оптического рефлектометра основано на явлении обратного рассеяния света в кабеле и на отражении света от скачков показателя преломления. Импульсы света, распространяясь по кабелю, испытывают отражения и затухания на неоднородностях кабеля и вследствие поглощения в среде [3].

С помощью рефлектометра определяется местонахождение дефектов и повреждений, измеряется уровень потерь сигнала в любой точке оптического кабеля. Все что нужно для работы с оптическим рефлектометром – это доступ к одному концу кабеля. Оптический рефлектометр производит тысячи измерений по всей длине кабеля. Точки с результатами измерений могут находиться друг от друга на расстоянии от 0,5 до 16 м. Эти точки выводятся на экран и образуют наклонную линию идущую слева направо и сверху вниз. При этом по горизонтальной оси графика откладывается расстояние, а по вертикальной – уровень сигнала. Выбрав с помощью подвижных курсоров две любые точки с результатами измерений можно определить расстояние между ними и разницу между уровнями сигнала в этих точках [4]. Посылая в кабель световой импульс и измеряя время его распространения и интенсивность отражения от точек, находящихся внутри кабеля, рефлектометр выводит на экран дисплея рефлектограмму «уровень отраженного сигнала в зависимости от расстояния».

Рефлектограмму можно проанализировать на месте, немедленно распечатать для создания документации о сети или сохранить на диске компьютера для более позднего анализа и сопоставлений. По такой рефлектограмме опытный оператор может точно определить конец кабеля, местонахождение оптических стыков и потери в них, а также полные потери в кабеле. В большинстве последних моделей рефлектометров предусмотрена возможность автоматического анализа полученных рефлектограмм, что упрощает обучение операторов. Для измерения характеристик оптического кабеля оптический рефлектометр использует явления релеевского рассеяния и френелевского отражения.

Релеевское рассеяние. При посылке светового импульса по кабелю часть импульса натывается на имеющиеся в стекле микроскопические частицы (которые называются «примесью») и рассеивается во всех направлениях. Это явление называется релеевским рассеянием. Часть световой энергии (около 0,0001%) рассеивается назад в направлении, противоположном направлению распространения импульса, – обратное рассеяние. Поскольку в процессе изготовления кабеля примеси распределяются равномерно по всему кабелю, явление рассеяния возникает по всей его длине [5].

На рисунке 1 изображено Релеевское рассеяние.

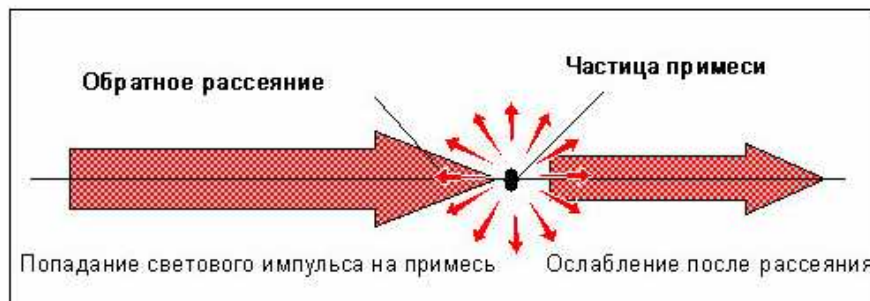


Рисунок 1. – Релеевское рассеяние

Релеевское рассеяние – основная причина потерь, имеющих место в кабеле. На более длинных световых волнах рассеяние меньше, чем на более коротких. Так, например свет на 1550 нм теряет из-за релеевского рассеяния от 0,2 до 0,3 (дБ/км) в то время как на 850 нм – от 4,0 до 6,0 (дБ/км). Имеющие

более высокую плотность примеси также увеличивают рассеяние и, следовательно, повышают уровень удельного затухания. Оптический рефлектометр может измерять уровни обратного рассеяния с большой точностью, используя эту способность для выявления незначительных изменений характеристик кабеля в любой его точке.

Релеевское рассеяние похоже на рассеивание частицами влаги луча света от карманного фонарика в ночном тумане. В густом тумане рассеивание будет сильнее, так как в воздухе больше частиц влаги. Туман виден потому, что частицы влаги рассеивают небольшое количество света по направлению к вам. Если туман не очень густой, луч света может распространяться на большое расстояние, но в густом тумане свет из-за эффекта рассеяния затухает довольно быстро. Частицы примесей в кабеле действуют, как частицы влаги в тумане, отражая при попадании на них света небольшое количество световой энергии назад к ее источнику [5].

Френелевское отражение. Всегда, когда свет, распространяющийся в каком-нибудь материале (например, в оптическом кабеле), попадает в материал с другой плотностью (например, в воздух) часть световой энергии (до 4%) отражается назад к источнику света, в то время как остальная световая энергия продолжает распространяться дальше. Резкие изменения плотности материала имеют место на концах кабеля у обрывов кабеля и иногда у оптических стыков. Количество отраженного света зависит от величины изменения плотности материала (которая характеризуется показателем преломления – более высокий показатель преломления означает большую плотность), а также от того угла, под которым свет падает на поверхность раздела между двумя материалами. Это явление называется френелевским отражением. Оно используется в оптическом рефлектометре для точного определения мест обрывов кабеля. Френелевское отражение напоминает ситуацию со светом карманного фонарика, падающим на оконное стекло. Большая часть света проходит через стекло, но какая-то его часть отражается назад к вам. От угла, под которым луч света падает на оконное стекло, зависит, куда попадет отраженный свет: назад в фонарик или же к вам в глаза [5]. Френелевское отражение изображено на рисунке 2.

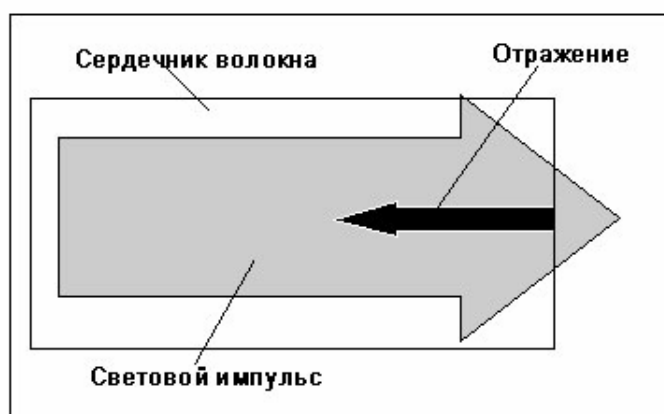


Рисунок 2. – Френелевское отражение

Заключение. В результате проведенных исследований рассмотрены виды отражений и рассеиваний в оптических кабелях. Рассмотрен принцип измерения характеристик оптических кабелей на основе определения интенсивности и времени задержки импульсных сигналов.

ЛИТЕРАТУРА

1. Оптические кабели [Электронный ресурс]. – Режим доступа: [http://ru.wikipedia.org/wiki/Оптические кабели](http://ru.wikipedia.org/wiki/Оптические_кабели). – Дата доступа: 25.04.2016.
2. Оптические кабели (ОК) [Электронный ресурс]. – Режим доступа: <http://victel.by/OK/>. Дата доступа: 26.04.2016.
3. Оптический рефлектометр [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/Оптический_рефлектометр. – Дата доступа: 29.04.2016.
4. Анализ и расшифровка рефлектограмм [Электронный ресурс]. – Режим доступа: <http://www.teleinfo.ru/seminar/232.htm>. – Дата доступа: 29.04.2016.
2. Физическая энциклопедия [Электронный ресурс]. – Режим доступа: http://www.femto.com.ua/articles/part_2/3477.html. – Дата доступа: 03.05.2016.

УДК 621.396

ИЗМЕРЕНИЕ ХАРАКТЕРИСТИК ОПТИЧЕСКИХ КАБЕЛЕЙ С ПОМОЩЬЮ РЕФЛЕКТОМЕТРОВ

Е.В. СМЕТАНА

(Представлено: канд. техн. наук, доц. В.Ф. ЯНУШКЕВИЧ)

Приведена функциональная схема устройства для измерения характеристик оптических кабелей. Рассмотрен принцип измерения характеристик оптических кабелей на основе применения передатчика световых сигналов со схемами стабилизации мощности и температуры. Схема оптического приемника позволяет измерять уровень сигнала в цифровом виде, а также исследовать характеристики оптического тракта с помощью осциллографа на выходе линейного усилителя.

Оптический рефлектометр работает по принципу радара – посылает в волокно короткий мощный импульс света и сразу начинает измерять все отражения, которые возникают при движении этого импульса вдоль волокна[1–3]. Как только зондирующий световой импульс доходит до любой неоднородности в волокне, например сварки, коннектора или повреждения, сразу же часть света отражается от этого места и начинает двигаться по волокну в обратную сторону – в направлении к рефлектометру, где она регистрируется фотоприемником. Но сам зондирующий импульс, хоть и немного ослабленный, продолжает свое движение по волокну, последовательно отражаясь от всех встречаемых на своем пути неоднородностей, пока не дойдет до конца волокна или до места его полного обрыва.

Разработка функциональной схемы устройства. Функциональная схема разрабатываемого прибора представлена на рисунке 1.

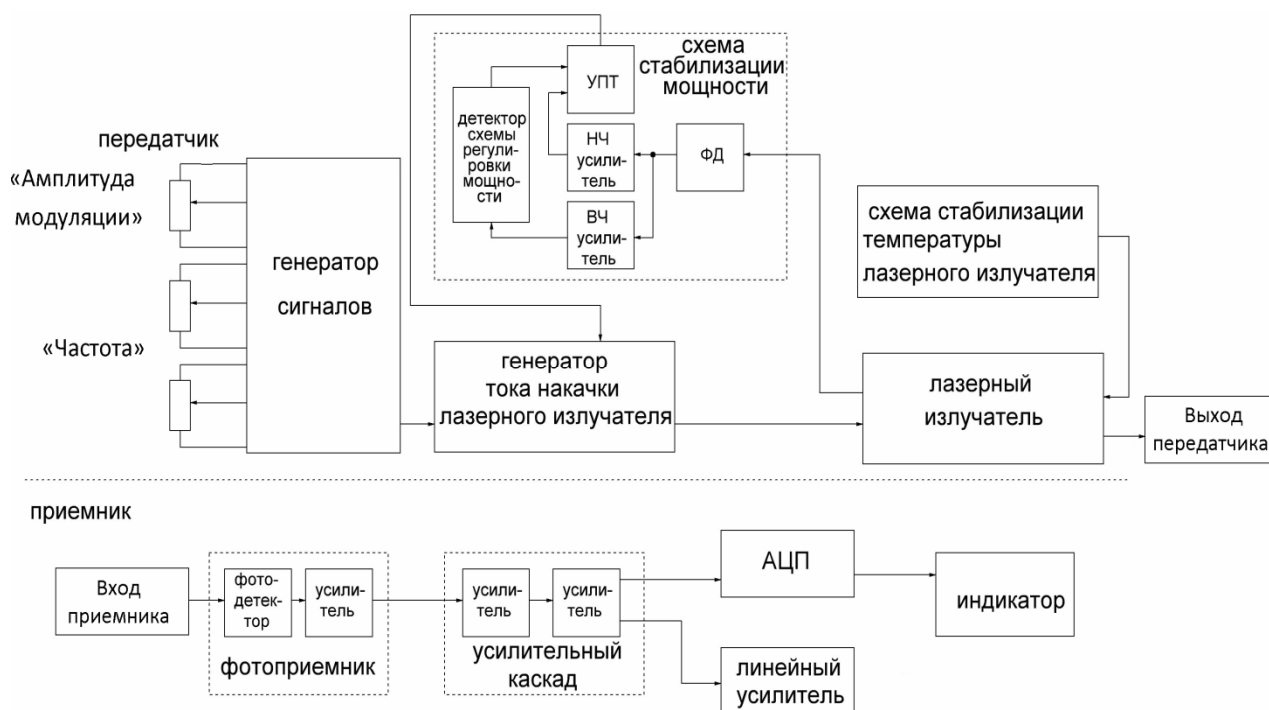


Рисунок 1. – Функциональная схема разрабатываемого устройства

Принцип работы прибора основан на анализе оптических импульсов, излучаемых рефлектометром в оптический кабель. Устройство состоит из приемника и передатчика.

Передатчик состоит из нескольких блоков. Необходимые импульсы получаются с помощью генератора сигналов. С помощью генератора накачки лазерного излучателя осуществляется накачка лазера, то есть процесс перекачки энергии внешнего источника в рабочую среду лазера. Поглощенная энергия переводит атомы рабочей среды в возбужденное состояние. Когда число атомов в возбужденном состоянии превышает количество атомов в основном состоянии, возникает инверсия населенности. В этом состоянии начинает действовать механизм вынужденного излучения и происходит излучение лазера или

же оптическое усиление. Стабилизатор мощности позволяет получить на выходе мощность, находящуюся в заданных пределах при значительно больших колебаниях входной мощности и сопротивления нагрузки. Схема стабилизации температуры обеспечивает требуемую температуру для работы устройства. Лазерный излучатель формирует электромагнитные волны оптического диапазона, и передает их в световод. Переданный с помощью лазерного излучателя, оптический сигнал попадает в фотоприемник – полупроводниковый прибор, регистрирующий оптическое излучение. Сигнал, принятый фотоприемником, усиливается с помощью схемы усилителя. Аналого-цифровой преобразователь преобразует оптический сигнал в электрический, для согласования с индикатором, с помощью которого отображаются полученные результаты измерения – рефлектограммы. Приемник измерителя снабжен линейным усилителем для контроля формы и амплитуды принимаемого сигнала [4–6].

Заключение

Разработана функциональная схема оптического рефлектометра для измерения характеристик оптических кабелей. Используется принцип измерения характеристик оптических кабелей на основе применения передатчика импульсных световых сигналов. Для улучшения технических характеристик оптического рефлектометра применены схемы стабилизации мощности и температуры лазерного излучателя. Схема оптического приемника позволяет измерять уровень сигнала в цифровом виде, а также исследовать характеристики оптического тракта с помощью осциллографа на выходе линейного усилителя.

ЛИТЕРАТУРА

1. Янушкевич, В.Ф. Устройства оптической обработки сигналов : метод. указания / В.Ф. Янушкевич. Новополоцк : ПГУ, 2002.
2. Оптические кабели [Электронный ресурс]. – Электронные данные. – Режим доступа: [http://ru.wikipedia.org/wiki/Оптические кабели](http://ru.wikipedia.org/wiki/Оптические_кабели). – Дата доступа: 25.04.2017.
3. Оптические кабели (ОК) [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://victel.by/OK/>. – Дата доступа: 26.04.2017.
4. Оптический рефлектометр [Электронный ресурс]. – Электронные данные. – Режим доступа: [http://ru.wikipedia.org/wiki/Оптический рефлектометр](http://ru.wikipedia.org/wiki/Оптический_рефлектометр). – Дата доступа: 29.04.2017.
5. Анализ и расшифровка рефлектограмм [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.teleinfo.ru/seminar/232.htm>. – Дата доступа: 29.04.2017.
6. Физическая энциклопедия [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.femto.com.ua/articles/part_2/3477.html. – Дата доступа: 03.05.2017.

УДК 621.371:550.837.6

ВЗАИМОДЕЙСТВИЕ РАДИОИМПУЛЬСНЫХ СИГНАЛОВ СО СРЕДОЙ
НАД УГЛЕВОДОРОДНЫМИ ЗАЛЕЖАМИ

В.Н. ЛЕВОЩЕНКО

(Представлено: канд. техн. наук, доц. В.Ф. ЯНУШКЕВИЧ)

Представлены результаты взаимодействия анизотропного слоя, расположенного над залежью, с ЭМВ в режиме радиоимпульсных сигналов. Выведен тензор диэлектрической проницаемости среды над УВЗ в режиме радиоимпульсных сигналов. Полученные результаты исследования могут быть исследованы в поисковой геофизике.

Актуальность рассматриваемых в настоящей работе задач заключается в модернизации существующих электромагнитных методов георазведки и разработке новых методов поиска, идентификации электрически неоднородных сред. Одной из разновидностей таких сред являются месторождения нефти и газа (углеводородов), являющиеся стратегическим видом полезных ископаемых и определяющие широкий спектр глобальных экологических вопросов современного общества [1].

Исследование режимов взаимодействия электромагнитных волн (ЭМВ) с углеводородными залежами (УВЗ) может быть использовано в поисковой геофизике для повышения точности и уровня достоверности методов обнаружения залежей нефти и газа.

Результаты анализа взаимодействия ЭМВ с УВЗ, экспериментальное исследование и интерпретация полученных исследований представлены во многих работах, например [2–5].

Вместе с тем для реализации новых методов представляет интерес анализ процесса взаимодействия ЭМВ с анизотропными средами (АС), образующихся над залежами и скоплениями углеводородов, в режиме радиоимпульсных сигналов [6].

Методика исследований. Проведем анализ компонентов диэлектрической проницаемости среды над углеводородами в режиме импульсного сигнала вида:

$$S(f) = \frac{U \cdot \tau_u}{2} \cdot \frac{\sin \frac{(\omega - \omega_0) \cdot \tau_u}{2}}{\frac{(\omega - \omega_0) \cdot \tau_u}{2}} + \frac{\sin \frac{(\omega + \omega_0) \cdot \tau_u}{2}}{\frac{(\omega + \omega_0) \cdot \tau_u}{2}},$$

где $\frac{U \cdot \tau_u}{2} = 1$ – амплитуда радиоимпульса; $\omega = 2 \cdot \pi \cdot f$ – частота радиоимпульса; ω_0 – частота несущего сигнала; τ_u – длительность импульса.

Взаимодействие ЭМВ со средой над углеводородами на трассе распространения радиоволн можно представить в виде режима наклонного падения плоской волны с вертикальной поляризацией в среде с параметрами $\epsilon_0, \mu_0, \delta_0$ на безграничную поверхность с анизотропным импедансом.

Выведем тензор диэлектрической проницаемости среды над УВЗ в режиме радиоимпульсных сигналов в следующем виде:

$$\dot{\epsilon} = \begin{vmatrix} \dot{\epsilon}_1 & -j\dot{\epsilon}_2 & 0 \\ j\dot{\epsilon}_2 & \dot{\epsilon}_1 & 0 \\ 0 & 0 & \dot{\epsilon}_3 \end{vmatrix} \quad (1)$$

Определение компонентов тензора диэлектрической проницаемости (1) основано на квазигидродинамическом приближении, в рамках которого уравнение движения электрона имеет вид:

$$m \frac{d\dot{\vartheta}}{dt} + v m \dot{\vartheta} = q \dot{E} + q \mu_0 \left[\dot{\vartheta}, \dot{H}_0 \right]. \quad (2)$$

В формуле (2) $m, q, \dot{v}$ – масса, заряд и скорость движения частиц; ν – частота столкновений электрона с тяжелыми частицами; μ_0 – магнитная проницаемость вакуума; $\vec{E}$ – напряженность электрического поля ЭМВ; $\dot{H}_0$ – напряженность магнитного поля Земли.

Координатное разложение напряженности электрического поля с учетом (2) позволяет записать компоненты тензора диэлектрической проницаемости в виде:

$$\left\{ \begin{aligned} \dot{\epsilon}_1 &= \epsilon_r \cdot F \left(1 + \frac{n\Omega_1}{w} \right) + \sum_{i=1}^2 \left\{ \begin{aligned} &\frac{F \cdot w_{ni}^2 \cdot (w+n\Omega_1) \cdot [w_{ni}^2 - (w+n\Omega_1)^2 - v_i^2]}{w \cdot [v_i^2 + w_{ni}^2 - (w+n\Omega_1)^2]^2 + 4(w+n\Omega_1)^2 \cdot v_i^2} - \\ &- jF \left[\frac{w_{ni}^2}{w} \cdot v_i \cdot \frac{(w+n\Omega_1)^2 + v_i^2 + w_{ni}^2}{[v_i^2 + w_{ni}^2 - (w+n\Omega_1)^2]^2 + 4(w+n\Omega_1)^2 \cdot v_i^2} + \frac{\delta_r F}{w\epsilon_0} \right] \end{aligned} \right\}, \\ \dot{\epsilon}_2 &= \sum_{i=1}^2 \left\{ \begin{aligned} &\frac{w_{ni}^2}{w} \cdot F \cdot w_{ni} \cdot \frac{v_i^2 + w_{ni}^2 - (w+n\Omega_1)^2}{[v_i^2 + w_{ni}^2 - (w+n\Omega_1)^2]^2 + 4(w+n\Omega_1)^2 \cdot v_i^2} - \\ &- 2j \frac{w_{ni}^2}{w} \cdot F \cdot w_{ni} \cdot v_i \cdot \frac{w+n\Omega_1}{[v_i^2 + w_{ni}^2 - (w+n\Omega_1)^2]^2 + 4(w+n\Omega_1)^2 \cdot v_i^2} \end{aligned} \right\}, \\ \dot{\epsilon}_3 &= \sum_{i=1}^2 \left\{ F \left(1 + \frac{n\Omega_1}{w} \right) \cdot \left[\epsilon_r - \frac{w_{ni}^2}{w} \cdot \frac{1}{(w+n\Omega_1)^2 + v_i^2} \right] - j \left[\frac{w_{ni}^2}{w} \cdot F \cdot v_i \cdot \frac{1}{(w+n\Omega_1)^2 + v_i^2} + \frac{\delta_2 F}{w\epsilon_0} \right] \right\}. \end{aligned} \right. \quad (3)$$

Здесь
$$F = \frac{\tau}{T} \cdot \sum_{n=-\infty}^{\infty} \frac{\sin(\pi \cdot n \cdot F_1 \cdot \tau)}{\pi \cdot n \cdot F_1 \cdot \tau};$$

$T, F_1; n$ – период, частота импульса и номер гармоники; $\Omega_1 = 2 \cdot \pi \cdot F_1$,

- гиротропная частота

$$\omega_{gi} = \frac{q_i \mu_0 H_0}{m};$$

- плазменная частота

$$\omega_{ni} = q_e \left(\frac{N_e}{m\epsilon_0} \right)^{\frac{1}{2}}.$$

Компоненты тензора (3) $\epsilon_1, \epsilon_2, \epsilon_3$, подлежащие исследованию в различных режимах взаимодействия среды с ЭМВ, несут информацию о свойствах анизотропной среды.

Заключение

Проведен анализ взаимодействия анизотропного слоя, расположенного над залежью, с ЭМВ в режиме радиоимпульсных сигналов. Выведен тензор диэлектрической проницаемости среды над УВЗ в режиме радиоимпульсных сигналов.

Результаты исследования могут быть исследованы в поисковой геофизике.

При этом следует отметить:

- тензор диэлектрической проницаемости среды над залежами нефти и газа в режиме радиоимпульсных сигналов может быть использован для определения электродинамических характеристик ис-

следуемого геопрофиля при вариации частот зондируемых сигналов, диэлектрических проницаемостей и удельных проводимостей сред в широком диапазоне;

- вариация характеристик зондирующих сигналов позволяет повысить информативность методов оконтуривания и выделения УВЗ.

ЛИТЕРАТУРА

1. Moskvichew, V.N. Interraction of electromagnetic waves (EMW) with anisotropic inclusion in communication line / V.N. Moskvichew // 9-th Microw. Conf. NICON – 91, Rydzyna, May 20–22. – 1991. – Vol. 1. – P. 240–244.
3. Гололобов, Д.В. Взаимодействие электромагнитных волн и углеводородных залежей / Д.В. Гололобов. – Минск : Бестпринт, 2009. – 185 с.
4. Гололобов, Д.В. Поверхностный импеданс среды над углеводородными залежами в режиме частотно-модулированных сигналов / Д.В. Гололобов, С.В. Калинин, В.Ф. Янушкевич // Весці НАН Беларусі. Сер. фіз.-тэхн. навук. – Минск, 2010. – № 4. – С. 98–101.
5. Asch T. and Morrison. Mapping and monitoring electrical resistivity with surface and surface electrode arrays // Geophysics. – 1989. – P. 235–244.
6. Гололобов, Д.В. Поиск месторождений углеводородных залежей радиоволновым методом / Д.В. Гололобов, В.Ф. Янушкевич // Приборы. Справ. журн. – М. : Научтехмаш, 1999. – № 10. – С. 25–27.
7. Янушкевич, В.Ф. Взаимодействие электромагнитных волн с анизотропными средами над углеводородными залежами в режиме радиоимпульсных сигналов / В.Ф. Янушкевич // Вестн. Полоц. гос. ун-та. Серия С, Фундаментальные науки. Физика. – 2016. – № 12. – С. 45–50.

УДК 621.371:550.837.6

МОДЕЛИРОВАНИЕ ВЗАИМОДЕЙСТВИЯ ИМПУЛЬСНЫХ СИГНАЛОВ СО СРЕДОЙ НАД УГЛЕВОДОРОДАМИ

В.Н. ЛЕВОЩЕНКО

(Представлено: канд. техн. наук, доц. В.Ф. ЯНУШКЕВИЧ)

Представлены результаты моделирования взаимодействия углеводородных залежей с ЭМВ в режиме радиоимпульсных сигналов. Проведен анализ комбинационных составляющих тензора диэлектрической проницаемости среды над УВЗ в режиме радиоимпульсных сигналов. Результаты исследования могут быть исследованы в поисковой геофизике.

В настоящее время для реализации новых методов георазведки представляет интерес исследование процесса взаимодействия радиоимпульсных электромагнитных волн (ЭМВ) со средой над залежами и скоплениями углеводородов, позволяющее оптимизировать частотные, поляризационные, амплитудные, модуляционные и другие параметры зондирующих сигналов, позволяющих повысить уровень идентификации углеводородов [1–4]. Цель настоящей работы состоит в обосновании выбора характеристик ЭМВ при использовании радиоимпульсных сигналов для разработки приборов поиска и идентификации углеводородных залежей (УВЗ).

Методика исследований. Проведем анализ компонентов диэлектрической проницаемости среды над углеводородами в режиме импульсного сигнала с параметрами: $\frac{U \cdot \tau_u}{2} = 1$ – амплитуда радиоимпульса; $\omega = 2 \cdot \pi \cdot f$ – частота радиоимпульса; ω_0 – частота несущего сигнала; τ_u – длительность импульса.

Взаимодействие ЭМВ со средой над углеводородами представим в виде наклонного падения плоской волны с вертикальной поляризацией в среде с параметрами $\epsilon_0, \mu_0, \delta_0$ на безграничную поверхность с анизотропным импедансом (рис. 1).

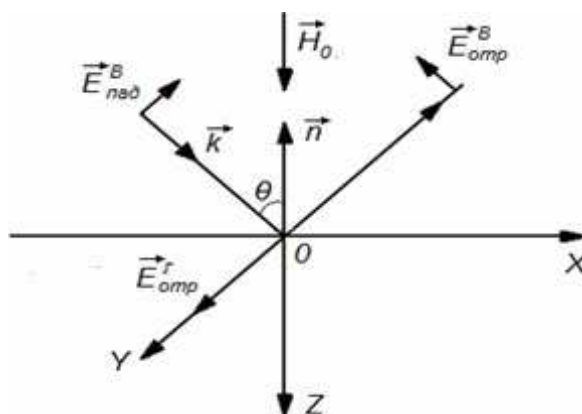


Рисунок 1. – Геометрия задачи для ЭМВ с вертикальной поляризацией поля

Проведем анализ тензора диэлектрической проницаемости среды над УВЗ в режиме радиоимпульсных сигналов в виде:

$$\hat{\epsilon} = \begin{vmatrix} \epsilon_1 & -j\epsilon_2 & 0 \\ j\epsilon_2 & \epsilon_1 & 0 \\ 0 & 0 & \epsilon_3 \end{vmatrix} \quad (1)$$

Компоненты тензора (1) $\epsilon_1, \epsilon_2, \epsilon_3$, подлежащие исследованию в различных режимах взаимодействия среды с ЭМВ, несут информацию о свойствах анизотропной среды.

П

редставляет интерес анализ частотных характеристик комбинационных составляющих

$$\begin{aligned}\dot{\mathbf{E}}_R(\omega) &= \dot{\mathbf{E}}_1 + \dot{\mathbf{E}}_2 = \text{Re } \mathbf{E}_R + j \text{Im } \mathbf{E}_R, \\ \dot{\mathbf{E}}_L(\omega) &= \dot{\mathbf{E}}_1 - \dot{\mathbf{E}}_2 = \text{Re } \mathbf{E}_L + j \text{Im } \mathbf{E}_L.\end{aligned}\quad (2)$$

Компоненты тензора диэлектрической проницаемости имеют вид[3]:

$$\left\{ \begin{aligned} \dot{\epsilon}_1 &= \epsilon_r \cdot F \left(1 + \frac{n\Omega_1}{w}\right) + \sum_{i=1}^2 \left\{ \frac{F \cdot w_{ni}^2 \cdot (w + n\Omega_1) \cdot [w_{fi}^2 - (w + n\Omega_1)^2 - v_i^2]}{w \cdot [v_i^2 + w_{fi}^2 - (w + n\Omega_1)^2]^2 + 4(w + n\Omega_1)^2 \cdot v_i^2} - \right. \\ &\quad \left. - jF \left[\frac{w_{ni}^2}{w} \cdot v_i \cdot \frac{(w + n\Omega_1)^2 + v_i^2 + w_{fi}^2}{[v_i^2 + w_{fi}^2 - (w + n\Omega_1)^2]^2 + 4(w + n\Omega_1)^2 \cdot v_i^2} + \frac{\delta_r F}{w\epsilon_0} \right] \right\}, \\ \dot{\epsilon}_2 &= \sum_{i=1}^2 \left\{ \frac{\frac{w_{ni}^2}{w} \cdot F \cdot w_{fi} \cdot \frac{v_i^2 + w_{fi}^2 - (w + n\Omega_1)^2}{[v_i^2 + w_{fi}^2 - (w + n\Omega_1)^2]^2 + 4(w + n\Omega_1)^2 \cdot v_i^2} - \right. \\ &\quad \left. - 2j \frac{w_{ni}^2}{w} \cdot F \cdot w_{fi} \cdot v_i \cdot \frac{w + n\Omega_1}{[v_i^2 + w_{fi}^2 - (w + n\Omega_1)^2]^2 + 4(w + n\Omega_1)^2 \cdot v_i^2} \right\}, \\ \dot{\epsilon}_3 &= \sum_{i=1}^2 \left\{ F \left(1 + \frac{n\Omega_1}{w}\right) \cdot \left[\epsilon_r - \frac{w_{ni}^2}{w} \cdot \frac{1}{(w + n\Omega_1)^2 + v_i^2} \right] - j \left[\frac{w_{ni}^2}{w} \cdot F \cdot v_i \cdot \frac{1}{(w + n\Omega_1)^2 + v_i^2} + \frac{\delta_2 F}{w\epsilon_0} \right] \right\}. \end{aligned} \right. \quad (4)$$

Здесь плазменная частота

$$\omega_{ni} = q_e \left(\frac{N_e}{m\epsilon_0} \right)^{\frac{1}{2}}.$$

$$F = \frac{\tau}{T} \cdot \sum_{n=-\infty}^{\infty} \frac{\sin(\pi \cdot n \cdot F_1 \cdot \tau)}{\pi \cdot n \cdot F_1 \cdot \tau},$$

T , F_1 , n – период, частота импульса и номер гармоники, $\Omega_1 = 2 \cdot \pi \cdot F_1$,
гиротропная частота

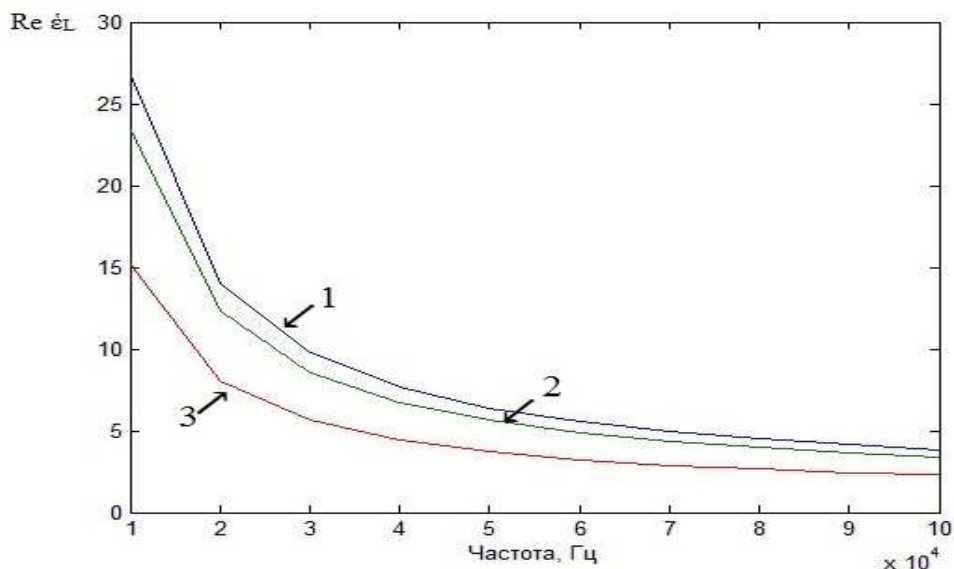
$$\omega_{fi} = \frac{q_i \mu_0 H_0}{m}.$$

Проведен анализ выражений (2) в диапазоне частот радиоимпульсных сигналов $F_i = (10^5 - 10^7)$ Гц, что обусловлено обеспечением глубин зондирования до 500 м для определения характеристик сред над залежами на больших глубинах по сравнению с «дневной» поверхностью [2].

Графики зависимостей приведены для частот радиоимпульса $1,5 \cdot 10^5$ Гц (рис.2), $5 \cdot 10^5$ Гц (рис. 3).

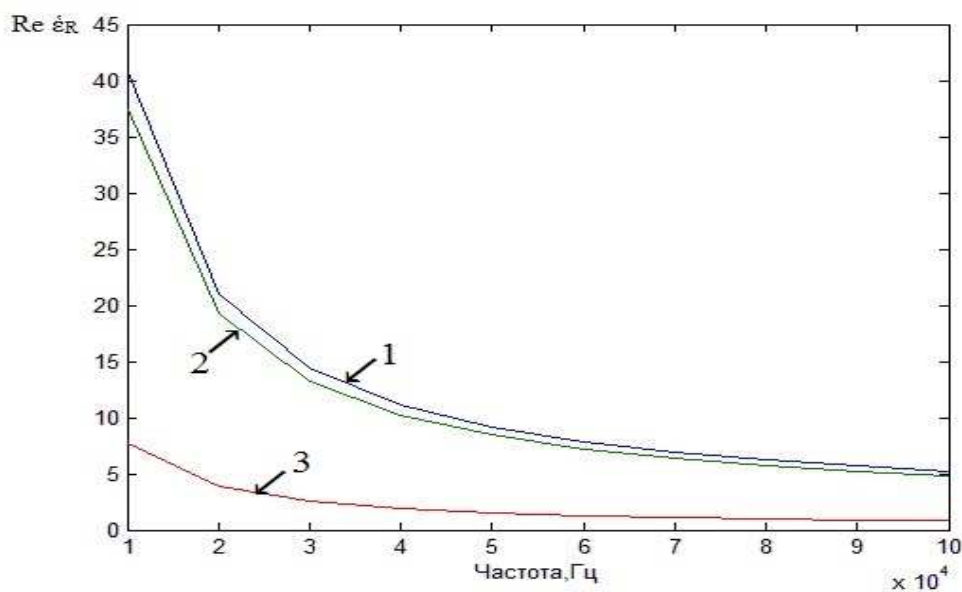
На практике встречается сложная картина распределения диэлектрической проницаемости среды над залежами, определяемая физико-геологическими свойствами пород-коллекторов:

- гранулометрическим и минералогическим составом пород;
- структурно-текстурными особенностями скелета;
- характером пористости и проницаемости;
- видом цементирующего вещества и механическими свойствами скелета;
- количеством пластовой воды в порах и ее минерализацией и т.д. [1].



1 – для $\epsilon_r = 25$; 2 – для $\epsilon_r = 20$; 3 – для $\epsilon_r = 2$

Рисунок 2. – Зависимости при $F_H = 1,5 \cdot 10^5$ Гц



1 – для $\epsilon_r = 25$; 2 – для $\epsilon_r = 20$; 3 – для $\epsilon_r = 2$

Рисунок 3. – Зависимости при $F_H = 5 \cdot 10^5$ Гц

Для повышения уровня достоверности определения границ УВЗ требуется обеспечение надежного метода поиска. Применение вспомогательных частот зондирования позволяет решить эту проблему.

Использование для зондирования частот импульса $1,5 \cdot 10^5$ и $5 \cdot 10^5$ Гц повышает информативность устройства поиска.

На частоте $1,5 \cdot 10^5$ Гц наблюдается различный характер изменения суммарной компоненты в зависимости от свойств диэлектрического наполнителя, а на частоте $5 \cdot 10^5$ Гц вещественная часть диэлектрической проницаемости ЭМВ с левой поляризацией отличается численными значениями, изменяясь по одинаковому закону.

Заключение

Результаты исследования могут быть исследованы в поисковой геофизике. При этом следует отметить следующее:

- распространение ЭМВ с правой и левой круговыми поляризациями повышает информативность методов оконтуривания и выделения УВЗ за счет различных закономерностей изменения электродинамических характеристик среды над залежью;

- методы поиска и идентификации углеводородов могут быть усовершенствованы за счет получения информации об объектах по нескольким каналам, что позволяет с достаточно высоким уровнем достоверности выделять их границы на фоне подстилающей среды.

ЛИТЕРАТУРА

1. Levashov, S.P. Electric-resonance sounding method and its application for, geological-geophysical and engineering-geological investigations / S.P. Levashov // 66nd EAGE Conference and Technical Exhibition, Paris, France, 7–10 June 2003 (CD-ROM Abstracts volume).
2. Янушкевич, В.Ф. Моделирование двухчастотного взаимодействия электромагнитных волн с углеводородной залежью / В.Ф. Янушкевич // Приборы. Справ. журн. – 1999. – № 10. – С. 27–29.
3. Взаимодействие электромагнитных волн с анизотропными средами над углеводородными залежами в режиме радиоимпульсных сигналов / В.Ф. Янушкевич [и др.] // Вестн. Полоц. гос. ун-т. Серия, С. Фундаментальные науки. – 2016. – № 12. – С. 45– 50.
4. Adamovskiy, E. Simulation of electromagnetic waves interaction with hydrocarbon deposits / E. Adamovskiy, V. Yanushkevich // 8 Junior researchers conference European and national dimension in research : In 3 Parts. – PSU, Novopolotsk, 2016. – Part 3: Technology. – P. 179–183.
5. Иванова, К.И. Способ геоэлектроразведки углеводородной залежи с использованием радиоимпульсных сигналов / К.И. Иванова, В.Ф. Янушкевич // Фундаментальные и прикладные исследования в современном мире : материалы XV Междунар. науч.-техн. конф., Санкт-Петербург, 4 окт. 2016 г. Т. 1. – СПб., 2016. – С. 107–111.

УДК 621.3.08

**КОНСТРУКЦИЯ ИСПЫТАТЕЛЬНОГО УСТРОЙСТВА
ДЛЯ ОПРЕДЕЛЕНИЯ УПРУГИХ СВОЙСТВ МАТЕРИАЛОВ****В.А. КРИШТОПА***(Представлено: канд. физ.-мат. наук, доц. С.А. ВАБИЩЕВИЧ)*

Рассматривается один из способов реализации устройства для изучения упругих свойств материалов. Линейные деформации и усилия определяются с помощью датчиков перемещения. Данные измерений обрабатываются микроконтроллером и позволяют проследить динамику упругих деформаций.

Экспериментальные методы исследования механических свойств материалов не теряют свою актуальность на протяжении уже достаточно длительного времени. Это обусловлено рядом причин, основными из которых являются [1; 2]:

- возможность исследования деформированного образца в целом, его отдельных узлов в реальных условиях эксплуатации, что является источником достоверной информации об эксплуатационной нагруженности элементов конструкции;

- необходимость использования экспериментальных результатов статических испытаний на различные типы деформации для построения диаграмм зависимости деформации от внешней нагрузки. Анализ указанных диаграмм позволяет рассчитать как основные механические характеристики материала (например, прочность и жесткость), так и его критические параметры: предел упругости, предел прочности и т.д.;

- наличие апробированных методов измерения деформации (тензометрия, поляризационно-оптический метод, метод голографической интерференции и др.).

Известно, что в современной измерительной аппаратуре часто используются бесконтактные методы, при которых измерение деформации основывается на взаимосвязи между деформацией и физическими параметрами материала (индуктивностью, емкостью, сопротивлением и т.п.).

Наиболее часто при этом используются методы исследования деформаций с помощью датчиков сопротивления. Точность полученных при этом измерений достаточна как для решения технических, так и научно-исследовательских задач.

Актуальность дальнейших исследований обусловлена, прежде всего, необходимостью разработки методик измерения, позволяющих получать результаты с минимальной погрешностью (менее нескольких процентов); наглядностью полученной информации и возможностью использования цифровых методов ее обработки. И, наконец, расширением элементной базы современных эластичных материалов, широко используемых в микро и нанoeлектронике (полимерные полупроводниковые материалы, пленки, нитевидные материалы и т.п.)

Основная цель работы состояла в разработке мобильного, компактного устройства, позволяющего проводить с минимальной погрешностью измерения механической одноосной деформации нитей, пленок и других тонких образцов, на основании которых можно было бы исследовать структуру образцов, рассчитывать критические механические параметры материала и характеристики его упругих свойств.

Схема устройства и методика измерений. Известно, что одноосное растяжение является наиболее простым с точки зрения экспериментального осуществления, сравнительно легко подвергается анализу, позволяет по результатам одного опыта определить характеристики таких важнейших механических свойств материалов, как упругость, прочность, статическая вязкость, удельную работу деформации и другие характеристики [1].

Чтобы измерять те или иные параметры материала при растяжении, следует измерять его удлинение при определенной нагрузке. Для этой цели было спроектировано устройство, принципиальная схема которого представлена на рисунке.

Испытательное устройство крепится неподвижными зажимами на основании. Образец 1 закреплен зажимами 2 и 3. Подвижный зажим 3 через эталонную пружину 5 связан с системой нагружения (электродвигатель). Подвижные элементы связаны с втулками, которые перемещаются по направляющим 4.

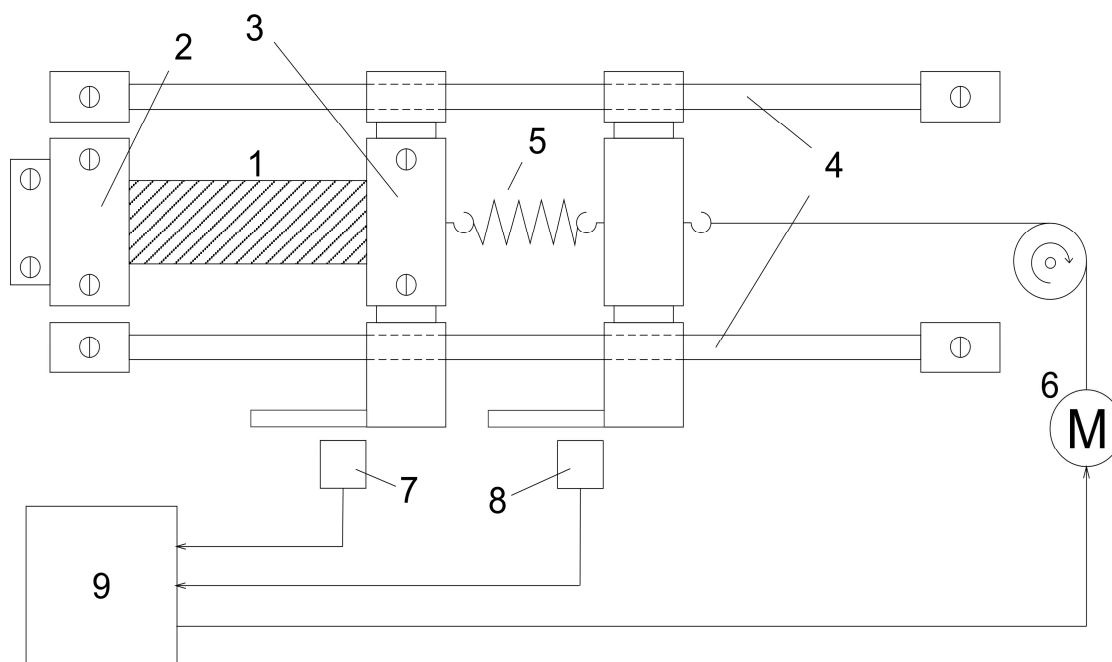
Датчики 7 и 8 преобразуют линейное перемещение в электрический сигнал. Датчиками выступают переменные резисторы. Датчик 7 служит для получения сведений о растяжении образца, а датчик 8 – для сведений о прилагаемом усилии.

Блок управления служит для предварительной обработки данных, управления нагружающим устройством и передачи информации на ПК для последующей обработки и хранения.

В качестве датчиков можно использовать датчики, основанные на любом из принципов измерения расстояния, будь то резистивные или оптические, емкостные или индуктивные датчики.

Оптимальными датчиками с точки зрения измерения нагрузки будут оптические, емкостные или индуктивные датчики, так как для измерения прилагаемого усилия такими датчиками можно исключить погрешность преодоления силы трения датчика.

Преимуществом измерения усилия с помощью пружины является то, что можно иметь сменный комплект таких пружин для вариации точности измерения усилия.



1 – образец; 2 – неподвижный зажим; 3 – подвижный зажим; 4 – направляющие;
5 – пружина; 6 – нагружающее устройство; 7, 8 – датчики перемещения; 9 – блок управления

Принципиальная схема испытательного устройства

Датчики 7 и 8 регистрируют линейное удлинение образца Δl_1 и пружины с образцом Δl_2 соответственно. Усилие, развиваемое пружиной, рассчитывается по удлинению пружины $F = k(\Delta l_2 - \Delta l_1)$, k – коэффициент жесткости пружины. Данные действия выполняет микроконтроллер, имеющий в своем составе аналого-цифровой преобразователь (АЦП), используемый для преобразования показаний датчиков в цифровой вид [3].

Затем рассчитывается относительное удлинение ϵ и соответствующее напряжение $\sigma(\epsilon)$.

Строится диаграмма растяжения и определяется модуль Юнга материала.

В качестве микроконтроллера следует использовать микроконтроллер Atmega8 или иной, имеющий в своем составе АЦП и универсальный асинхронный приемо-передатчик UART. В качестве альтернативы можно использовать микроконтроллер без АЦП, но тогда придется подключать внешний АЦП, что позволит, с одной стороны, увеличить чувствительность при использовании АЦП с большим количеством разрядов, чем штатный, но с другой стороны, стоимость такого изделия значительно вырастет.

Для увеличения скорости обмена информацией микроконтроллера с компьютером целесообразно использовать микроконтроллеры с аппаратной поддержкой USB интерфейса, например AT90USB. Такой подход чреват сложностями в написании необходимых драйверов, но является наиболее перспективным [4].

Для приема и обработки данных с микроконтроллера можно воспользоваться оконным приложением, написанным, к примеру, в среде разработки VisualStudio.

При нажатии оператором кнопки запуска приложение должно запускать нагружающее устройство и принимать данные, регистрируемые датчиками.

В соответствии с принятыми данными, программа строит графики в реальном времени, согласно которым можно судить об упругих свойствах испытываемого материала. Графики сохраняются в формате растрового изображения или таблицы значений по желанию оператора. Также можно воспользоваться одной из функций Matlab – InstrumentControlToolbox, что позволяет строить матрицы на основе полученных данных для их последующей обработки.

Заключение

Представленное устройство предназначено для изучения упругих свойств материалов при линейных деформациях в автоматическом режиме. Применение электронных датчиков позволяет изучать динамику процесса деформации с различными скоростями и получать более достоверные результаты, по сравнению с обычными механическими датчиками.

ЛИТЕРАТУРА

1. Остаточные напряжения в деформируемых твердых телах / Г.Н. Чернышев [и др.]. – М.: Наука, 1996. – 231 с.
2. Феодосьев, В.И. Сопротивление материалов : учебник для вузов / В.И. Феодосьев. – 16-е изд., испр. – М. : Изд-во МГТУ им. Н.Э. Баумана, 2016. – 543 с.
3. Трамперт, В. Измерение, управление и регулирование с помощью AVR-микроконтроллеров : пер. с нем. – К. : МК-Пресс, 2006. – 208 с.
4. Бондаренко, Д.Н. Встраиваемые микроконтроллеры AVR-8 : учеб. пособие по дисциплинам «микропроцессорные устройства» и «Компьютерная и микропроцессорная техника» / Д.Н. Бондаренко. – СПб., 2014. – 193 с.

УДК 621.3.08

ДАТЧИК ЛИНЕЙНОГО ПЕРЕМЕЩЕНИЯ РЕЗИСТИВНОГО ТИПА

В.А. КРИШТОПА

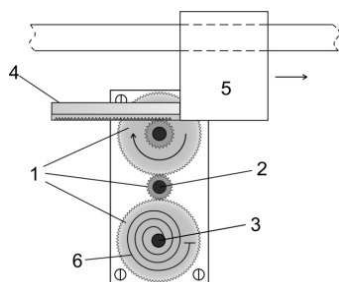
(Представлено: канд. физ.-мат. наук, доц. С.А. ВАБИЩЕВИЧ)

Рассматривается один из способов преобразования линейно-поступательного движения в электрический сигнал. В качестве датчика применены потенциометры. Данный прибор позволяет измерять линейные перемещения с микрометрической точностью. Измеренные значения предварительно обрабатываются микроконтроллером, после чего передаются на персональный компьютер для последующей обработки и извлечения информации.

Электронные датчики являются важной составляющей в автоматизации извлечения данных. Касательно испытательных стендов, с помощью электронных устройств можно увеличить количество измеряемых параметров образцов, а также увеличить разрешающую способность по сравнению с механическими датчиками. Применение электронных датчиков упрощает изучение динамики процесса, а также позволяет получать более достоверные данные, нежели с применением обычных механических датчиков.

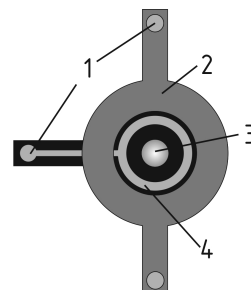
Для измерения динамики растяжения пленки следует извлекать два параметра на протяжении времени испытания ее механических свойств. Этими параметрами являются удлинение и сила, приложенная для этого удлинения. Чтобы измерять удлинение, достаточно измерять расстояние, на которое растягивается пленка. Для измерения усилия, можно воспользоваться динамометром. В этом случае, при применении пружины, нужно также знать, ее удлинение, а затем, пользуясь законом Гука, находить усилие, приложенное для ее растяжения. Менее дорогостоящим методом будет преобразование линейного перемещения в электрический сигнал посредством редуктора и переменного сопротивления. Метод заключается в воздействии линейного привода на шестерню, вращающую вал потенциометра. Более подробная разрабатываемая конструкция приведена на рисунке 1.

Принцип действия такого датчика заключается в том, что линейный привод 4, приводимый в действие перемещением каретки 5, заставляет вращаться связанные шестерни 1, тем самым приводя в движение ручки потенциометров 2 и 3. Пружина 6 требуется для создания натяжения для прижатия зубьев шестерен друг к другу, уменьшая тем самым люфты механизма. Особенность данного датчика заключается в том, что для повышения точности, в нем применен, помимо обычного потенциометра 3, со стандартным углом вращения, равным 300 градусов, и линейной зависимостью, потенциометр с бесконечным углом вращения. Схема такого сопротивления приведена на рисунке 2.



1 – шестерни; 2 – многооборотный потенциометр;
3 – потенциометр; 4 – линейный привод;
5 – каретка; 6 – пружина

Рисунок 1. – Механизм датчика



1 – контакты; 2 – резистивная поверхность;
3 – вал; 4 – контактная дорожка

Рисунок 2. – Конструкция многооборотного потенциометра

Преимуществом данной конструкции является то, что поверхность резистивного материала 2 равномерна и не имеет переходов между контактами 1, за счет чего удастся избавиться от «скачков» напряжения при переходе из максимального положения в минимальное. Однако судить о том, в какую сторону крутится вал 3 потенциометра, невозможно без применения дополнительных мер. Именно поэтому в конструкции применяется дополнительный потенциометр, который менее точно показывает измеряемое расстояние, но с точностью знает, в каком направлении идет движение.

Для преобразования аналогового сигнала датчика в цифровую форму следует воспользоваться аналого-цифровым преобразователем (АЦП).

Цифровое значение кода АЦП определяется соотношением [1]:

$$АЦП = \frac{U_{Bx} \cdot 2^n}{U_{REF}},$$

где U_{Bx} – напряжение, снимаемое с потенциометра; U_{REF} – опорное напряжение АЦП; n – разрядность АЦП. Опорное напряжение АЦП и напряжение, приложенное к крайнему выводу потенциометра, равно

$$U_{Bx} = U_{REF} \frac{\alpha}{\alpha_{\max}},$$

где α – угол поворота потенциометра, $\alpha_{\max}$ – максимальный угол поворота потенциометра. Таким образом, цифровое значение кода АЦП прямо пропорционально зависит от угла поворота потенциометра:

$$АЦП = \frac{\alpha \cdot 2^n}{\alpha_{\max}}.$$

Из этого выражения следует, что минимальное измеряемое значение угла, то есть при единичном дискретном значении АЦП будет равным:

$$\alpha = \frac{\alpha_{\max}}{2^n}.$$

Приняв разрядность АЦП, равную 10 битам, и стандартное максимальное значение поворота потенциометра, равную 300 градусам, минимальный угол поворота составляет:

$$\alpha_{\min} = \frac{300^\circ}{1024} \approx 0,3^\circ.$$

Длина дуги при этом, будет зависеть от радиуса шестерни, как:

$$l_{\min} = \frac{\pi}{180^\circ} \cdot 0,3^\circ \cdot r = 0,00525 \cdot r.$$

Так, при радиусе шестерни, передающей движение на ось потенциометра, равной 5 мм, приращение единицы кода будет равняться:

$$l_{\min} = 0,00525 \cdot 5 \text{ мм} = 0,02625 \text{ мм}.$$

Цена деления такого потенциометра составляет 26,25 мкм, а длина шкалы определяется, как:

$$l_{\max} = \frac{\pi}{180^\circ} \cdot 300^\circ \cdot 5 \text{ мм} = 26,25 \text{ мм}.$$

Совместное применение такого потенциометра с многооборотным позволяет, не уменьшая длину шкалы, увеличить точность прибора в удвоенное передаточное число раз. Это наглядно видно на графике, показанном на рисунке 3.

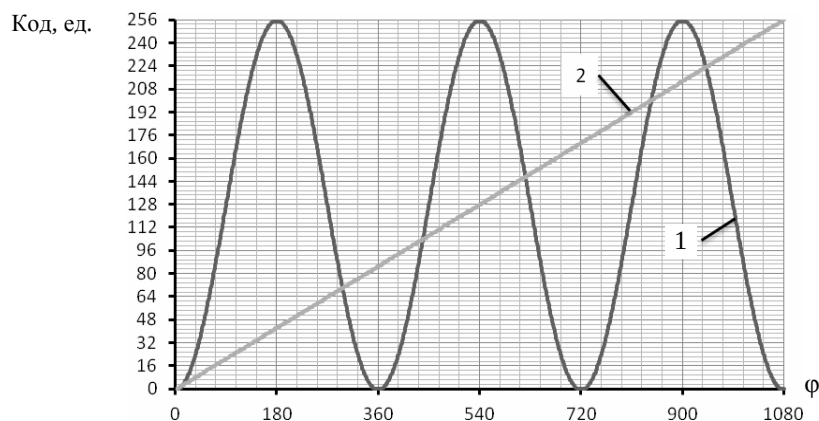


Рисунок 3. – Зависимость выходного кода АЦП от угла поворота многооборотного датчика

Высокая точность достигается путем увеличения крутизны датчика в целом, которая определяется крутизной многооборотного датчика. Крутизна, в свою очередь, зависит не только от передаточного числа, но и от конструкции многооборотного датчика. Так, при передаточном числе, равном единице, крутизна многооборотного датчика будет в 2 раза выше, чем у обычного потенциометра.

На графике, представленном на рисунке 3, за 1080° поворота многооборотного датчика, обычный потенциометр делает оборот на 300°, передаточное число равняется:

$$k = \frac{1080^\circ}{300^\circ} = 3,6.$$

При этом, учитывая, что крутизна данного потенциометра в 2 раза выше, чем обычного, общая разрешающая способность равняется 3,6 мкм:

$$I'_{\text{мин}} = \frac{I_{\text{мин}}}{2 \cdot 3,6} = \frac{0,02625 \text{ мм}}{7,2} = 0,0036 \text{ мм}.$$

Стоит учитывать, что это идеальная модель, ведь при перемещении щеток потенциометра возникают шумы порядка 50 мВ [2]. Электрическая схема для снятия показаний с подобных датчиков приведена на рисунке 4.

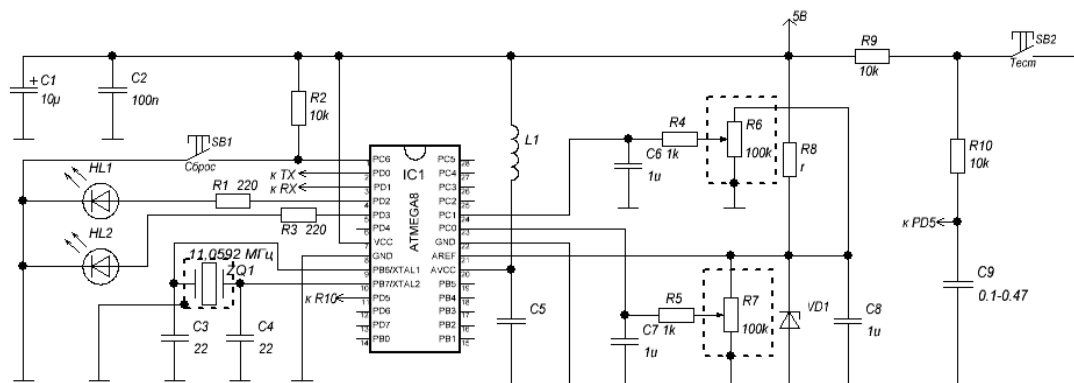


Рисунок 4. – Схема для снятия показаний датчиков

Схема основана на широко используемом микроконтроллере atmega8 [3], имеющем в своем составе 8- и 10-разрядный АЦП. Потенциометры R6 и R7 являются датчиковыми формирователями сигнала, который проходя через антиалиасинговые фильтры R4C6 и R5C7, попадает на мультиплексированный вход АЦП [4]. Цепочка L1C5 уменьшает влияние шумов и подводит опорное напряжение АЦП к микроконтроллеру. Дополнительно питание микроконтроллера стабилизируется стабилитроном VD1. Микроконтроллер тактируется внешним кварцевым резонатором с частотой, позволяющей свести к минимуму ошибки при передаче информации по протоколу RS-232. Микроконтроллер подключается через эмулятор СО Мпорта к ПК.

Заключение

Представленный датчик предназначен для измерения микроперемещений. Применение потенциометров позволяет удешевить и упростить конструкцию. Применение электронной схемы, позволяющей сохранять и обрабатывать результаты, позволяет изучать динамику процесса измерения.

ЛИТЕРАТУРА

1. Микросхемы АЦП и ЦАП : справочник. – М. : Издат. дом «Додэка-XXI», 2005. – 416 с. – (Интегральные микросхемы).
2. Марченко, А.Н. Переменные резисторы / А.Н. Марченко. – М. : Энергия, 1980. – 72 с. – (Массовая радиобиблиотека). – Вып. 1017.
3. Трамперт, В. AVR-RISC микроконтроллеры / В. Трамперт ; пер. с нем. – К. : «МК-Пресс». – 2006. – 464 с.
4. Рюмик, С.М. 1000 и одна микроконтроллерная схема / С.М. Рюмик. – М. : Додэка-XXI, 2010. – 356 с.

УДК 62.503.55

МОДЕРНИЗАЦИЯ СТАНКА РЯДОВОЙ НАМОТКИ РЯД-01

С.В. ПАСТИКОВ

(Представлено: С.Н. АБРАМЕНКО)

Рассмотрен процесс модернизации станка намотки Ряд01 путем замены электронной части управления на более надежную и современную электронную базу.

Постановка задачи. Станок Ряд01 (рисунок 1) на данный момент имеет достаточно устаревшую электронную часть управления, что увеличивает возможность появления различных неисправностей. Но механическая часть вполне пригодна.

Учитывая подобную ситуацию и потенциальный спрос предприятий региона на его модернизацию, принято решение полной замены электронного блока управления.



Рисунок 1. – Станок Ряд01

Основные задачи:

1. Заменить электронную схему управления и тип двигателя.
2. Использовать современные электронные составляющие для замены схемы счета и индикации.

Разработка структуры устройства. Существует два типа двигателей:

- 1) двигатель постоянного тока;
- 2) двигатель переменного тока (асинхронный).

В рассматриваемой ситуации перспективным является второй тип двигателя.

Асинхронные двигатели просты и надежны, но имеют недостаток – трудность регулировки частоты вращения. Учитывая появления частотных преобразователей, это проблема становится неактуальной.

Выбирая асинхронный двигатель, будем отталкиваться от параметров станка.

Скорость намотки:

- I ступень – от 0 до 8100 об/мин;
- II ступень – от 0 до 1380 об/мин;
- дополнительный привод – от 0 до 300 об/мин;
- потребляемая мощность – не более 0,7 кВт.

Следовательно, для нас оптимальный вариант – двигатель мощностью 750 Вт.

Для задания частоты вращения будем использовать частотный преобразователь Omron MX2.

Схема подключения представлена на рисунке 2.

Выбор схемы управления

При выборе схемы управления для нас были актуальны следующие задачи:

- 1) простота использования;
- 2) минимальные габариты;
- 3) высокий функционал.

Принимая во внимания все вышеперечисленные условия, предпочтение отдано торговой марке аппаратно-программных средств «Arduino» (рис. 3).

Старт.

Запуск станка с установленными начальными параметрами. Работа в этом режиме продолжается до тех пор, пока не будет набрано установленное количество витков.



Рисунок 4. – Матричная клавиатура

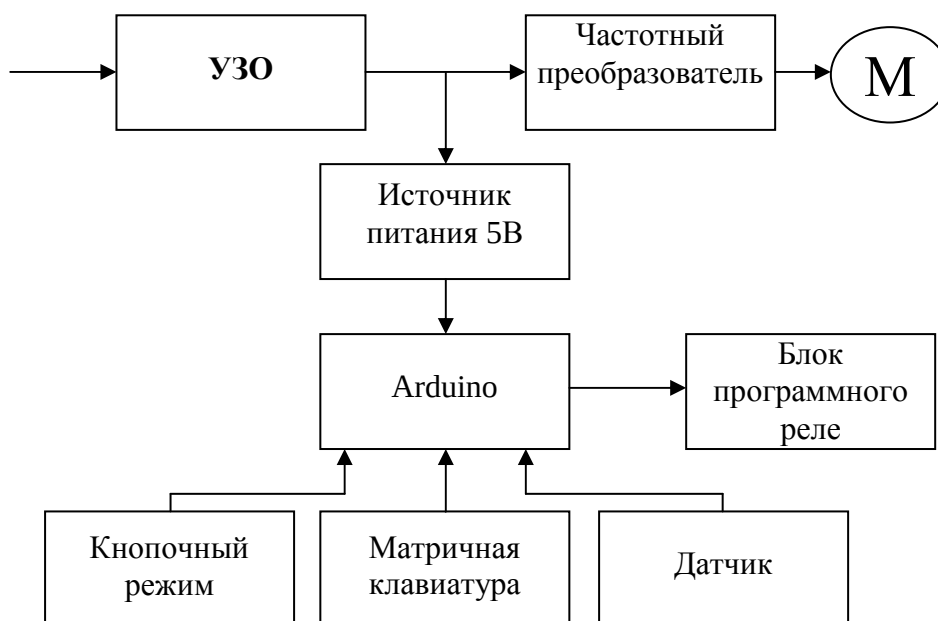


Рисунок 5. – Структурная схема

Заключение

В ходе дальнейших работ будет реализована доработка электрической схемы. Разработанное программное обеспечение будет полноценно протестировано при различных ситуациях. Решив поставленные задачи, будет осуществлен пробный запуск всей конструкции. Это поможет нам наглядно проанализировать ход работы станка с использованием нашей модификации.

ЛИТЕРАТУРА

1. Omron MX2 первый запуск.
2. Технические данные [Электронный ресурс]. – Режим доступа: <http://www.osnastka-market.com.ua/kinds/prod3/gyad01m/>. – Дата доступа: 12.06.2016.

УДК 621.371.39

ПРОЕКТИРОВАНИЕ GSM АНТЕННЫ

А.О. СТАНОВОЙ

(Представлено: канд. техн. наук, доц. В.Ф. ЯНУШКЕВИЧ)

Представлены результаты проектирования GSM антенн, использование которых в радиотехнических системах позволяет повысить возможности систем радиосвязи, радиолокации, а также обеспечить высокую помехозащищенность каналов связи, повысить точность при оценке взаимной ориентации движущихся объектов. Полученные результаты могут быть использованы в радиотехнических системах различного назначения, где требуются высокоэффективные антенны с улучшенными техническими характеристиками.

Актуальность представляемой работы определяется возросшими требованиями к техническим характеристикам GSM антенн.

Форма, размеры и конструкция антенн и разнообразны и зависят от длины излучаемых или принимаемых волн и назначения антенны.

Применяются антенны в виде отрезка провода, комбинаций из таких отрезков, отражающих металлических зеркал различной конфигурации, полостей с металлическими стенками, в которых вырезаны щели (щелевая антенна), спиралей из металлических проводов и другие.

GSM антенна – наиболее распространенная антенна, имеющая достаточно узкую диаграмму направленности в горизонтальной и вертикальной плоскостях. Антенна состоит из набора элементов:

- одного вибратора, к которому подводится и снимается сигнал (активный вибратор);
- нескольких пассивных вибраторов, один из которых является рефлектором, а остальные работают в режиме директоров.

Задача рефлектора – ослабить излучение в заднем, не рабочем, направлении.

Задача директора – усилить сигнал в нужном направлении.

Все вибраторы параллельны и лежат в одной плоскости, эта плоскость и определяет поляризацию антенны. Часто такую антенну называют антенной типа «волновой канал», или антенной «Уда – Яги».

Расчет антенны позволяет получить лишь приближенные результаты, так как необходимо учитывать много взаимосвязанных факторов (длину и толщину вибраторов, расстояния между ними), и связан с громоздкими вычислениями. Поэтому чаще всего конструируют антенну, подбирая все их размеры экспериментальным путем, придерживаясь следующих правил:

1) по заданной рабочей волне λ рассчитывают и устанавливают длину активного вибратора $2l$ с учетом укорочения Δl ;

2) устанавливают длину рефлектора $2l_p$ на 5–10% больше длины активного вибратора, а расстояние между вибраторами порядка $0,2\lambda$;

3) с помощью индикатора поля измеряют напряженность поля E_o (т.е. в сторону активного вибратора) и E_{180} (в сторону рефлектора), и регулировкой расстояния dr между вибраторами и длины рефлектора добиваются минимального отношения E_{180} / E_o . Для системы из двух вибраторов;

4) добавляют директор, длину которого $2l_q$ устанавливают короче на 10–15% длины активного, а расстояние a_d до последнего – порядка $0,2\lambda$, и регулировкой указанных размеров добиваются минимального отношения E_{180} / E_o . Для системы из трех вибраторов (при этом может потребоваться незначительная подстройка рефлектора);

5) добавляют последовательно второй, третий и т.д. директоры и производят аналогичные регулировки.

Форма диаграммы направленности (ДН), коэффициента стоячей волны (КСВ) и коэффициента отражения (S11) рассчитаны для частоты от 1,92 до 2,17 ГГц (рис. 1–3), выбран материал – медь.

Антенна имеет направленное излучение в направлении 0° , в направлении -180° наблюдается обратное малое излучение ДН, как показано на рисунке 1.

У антенны наблюдаются хорошие согласующие свойства на отрезке частот от 1,99 до 2,17 ГГц (рисунок 2).

Величина коэффициента отражения от входа уменьшается от 0 дБ на частоте от 1,92 до –10,4 дБ на частоте 2,17 ГГц, на частоте 2,045 ГГц коэффициент отражения равен –6 дБ.

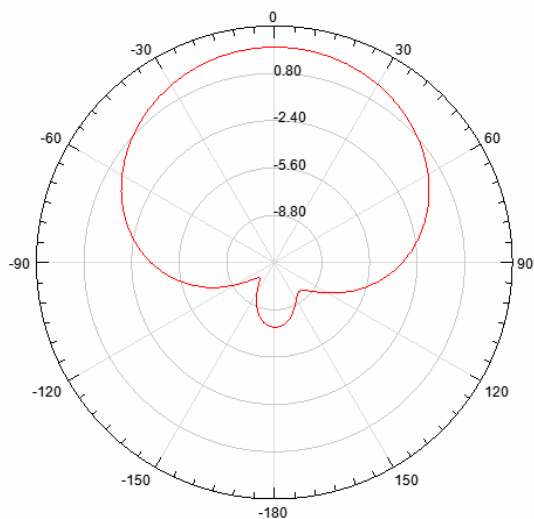


Рисунок 1. – Диаграмма направленности антенны

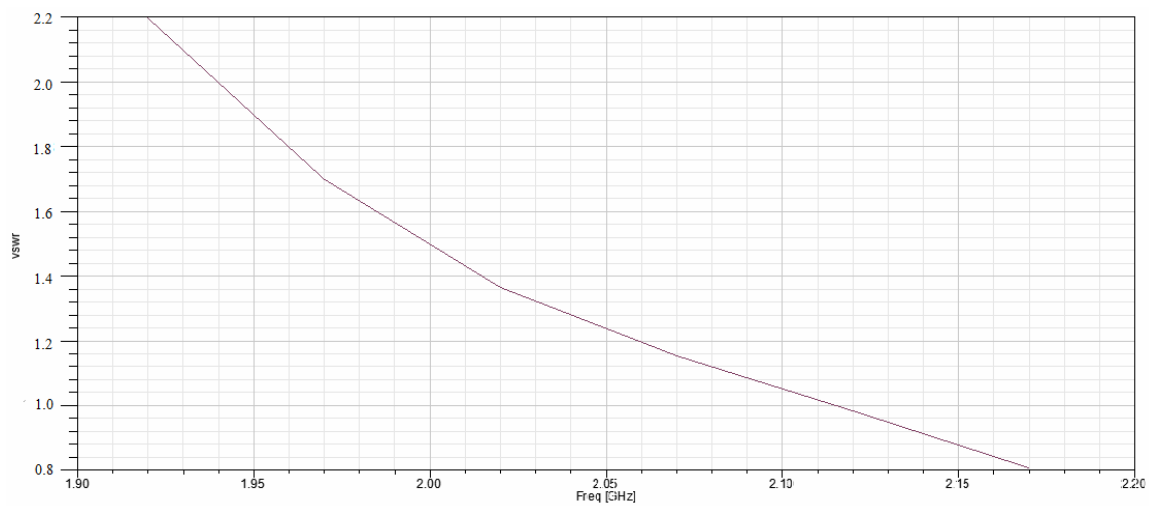


Рисунок 2. – Коэффициент стоячей волны антенны

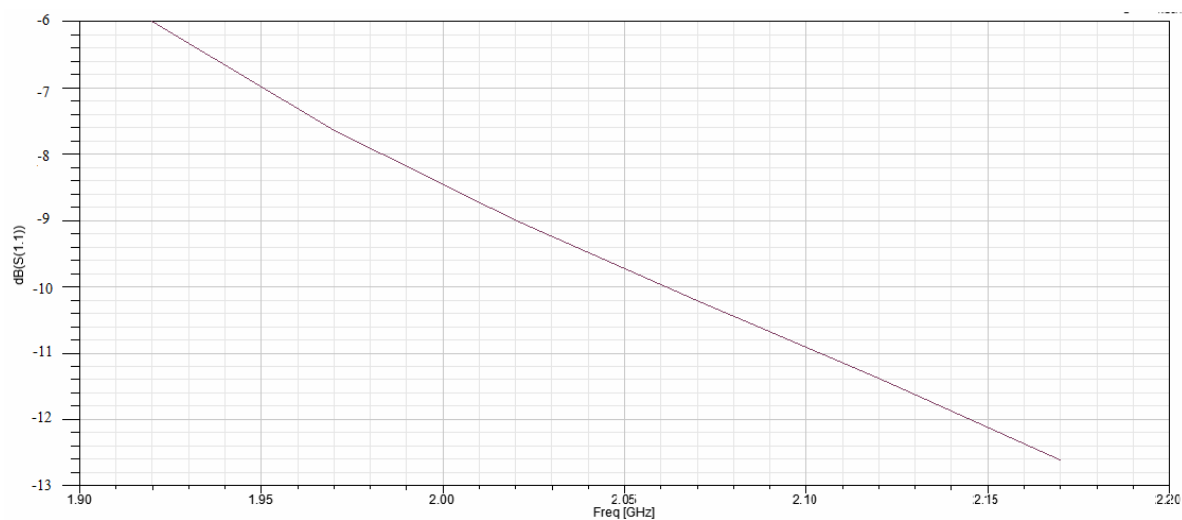


Рисунок 3. – S11 параметр антенны

Заключение

Результаты исследования могут быть использованы в радиотехнических системах различного назначения, где требуются высокоэффективные антенны с улучшенными техническими характеристиками.

В данной работе представлены результаты проектирования GSM антенны.

Проведено исследование таких характеристик антенны, как:

- диаграмма направленности;
- коэффициент стоячей волны;
- коэффициент отражения S_{11} , определяемый по входу антенны.

ЛИТЕРАТУРА

1. Юрцев, О.А. Антенны бегущей волны, антенные решетки, антенны коротких, средних и длинных волн : в 3 ч / О.А. Юрцев. – Минск : БГУИР, 2001. – Ч. 3 : Методическое пособие по курсу «Антенны и устройства СВЧ» для студентов специальности «Радиотехника» – 72 с.
2. Янушкевич, В.Ф. Методические указания к выполнению курсовой работы по дисциплине «Устройства СВЧ и антенны» для студентов специальности Т.09.01.00 / В.Ф. Янушкевич. – Новополоцк : ПГУ 1997.
3. Бова, Н.Т. Антенны и устройства СВЧ / Н.Т. Бова, Г.Б. Резников. – 2-е изд., перераб. и доп. – Киев : Вища школа. Головное изд-во, 1982. – 278 с.

УДК 519.6

**АНАЛИЗ ЭФФЕКТИВНОСТИ ИСПОЛЬЗОВАНИЯ МЕТОДОВ РУНГЕ – КУТТЫ
РАЗНОГО ПОРЯДКА ДЛЯ РЕШЕНИЯ ЗАДАЧ СТРОИТЕЛЬНОЙ МЕХАНИКИ****А.Н. ВОЛКОВ, Ю.В. КОШЕЛЕВА**
(Представлено: Т.М. ГЛУХОВА)

Исследуются методы решения дифференциальных уравнений в области строительной механики. Проводится анализ эффективности методов. Выделены плюсы и минусы функциональных возможностей тех или иных методов, определены их недостатки и достоинства.

Многие задачи строительной механики, например, задачи изгиба стержней или балок, лежащих на упругом основании, задачи расчета оболочек вращения и многие другие сводятся к решению начальных и краевых задач для обыкновенных дифференциальных уравнений и систем таких уравнений. Для их решения имеются простые и эффективные методы, такие как метод Эйлера, модифицированный метод Эйлера, метод Рунге – Кутты третьего порядка, метод Рунге – Кутты четвертого порядка, метод Рунге – Кутты – Фельберга и другие. Цель работы проанализировать данные методы на эффективность использования и сравнить их быстродействия с использованием ЭВМ.

Метод Эйлера

Метод Эйлера относится к численным методам, дающим решение в виде таблицы приближенных значений искомой функции $y(x)$. Он является сравнительно грубым и применяется в основном для ориентировочных расчетов. Однако идеи, положенные в основу метода Эйлера, являются исходными для ряда других методов. Метод Эйлера является самым простым, для реализации каждое последующее приближение рассчитывается по формуле:

$$y_{n+1} = y_n + hk_1;$$

$$y_{n+1} = x_n + h,$$

$$k = f(x_n, y_n).$$

Идея методов Эйлера и Рунге – Кутты состоит в том, чтобы заменить фрагмент графика ломаной линией (кривой Эйлера), которая имеет направление, совпадающее с направлением той интегральной кривой.

Существенным недостатком простого метода Эйлера является слишком большая погрешность, при этом легко заметить, что погрешность имеет тенденцию накапливаться.

Модифицированный метод Эйлера

Существуют модифицированные методы Эйлера. К примеру, метод средних точек, рассчитывающий приближения по следующим формулам:

$$y_{n+1} = y_n + hk_2;$$

$$y_{n+1} = x_n + h;$$

$$k_1 = f(x_n, y_n);$$

$$k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{hk_1}{2}\right).$$

Однако значительного уменьшения погрешности при применении данных методов не наблюдается.

Методы Рунге – Кутты

Методы типа Рунге – Кутты являются наиболее популярными одношаговыми методами численного решения обыкновенных дифференциальных уравнений (ОДУ) в настоящее время. Рассмотрим несколько из этих методов.

Метод Рунге – Кутты третьего порядка

Для увеличения точности решения применяют методы Рунге – Кутты разных порядков. Например, метод 3-го порядка требует на каждом шаге вычислять по 3 значения функции $f(x)$ (для каждого коэффициента k):

$$y_{n+1} = y_n + h\left(\frac{k_1}{6} + \frac{2k_2}{3} + \frac{k_3}{6}\right),$$

$$x_{n+1} = x_n + h.$$

$$k = f(x_n, y_n).$$

$$k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{hk_1}{2}\right).$$

$$k_3 = f(x_n + hy_n + 2hk_2 - hk_1).$$

Метод Рунге – Кутты четвертого порядка

Наиболее распространенным (классическим) считается метод Рунге – Кутты 4-го порядка, где значение функции потребуется рассчитать 4 раза на каждом шаге расчета.

Расчет каждого приближения вычисляется по формулам:

$$y_{n+1} = y_n + h\left(\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{6} + \frac{k_4}{6}\right),$$

$$x_{n+1} = x_n + h.$$

$$k_1 = f(x_n, y_n).$$

$$k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{hk_1}{2}\right).$$

$$k_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{hk_2}{2}\right).$$

$$k_4 = f(x_n + hy_n + hk_3).$$

Метод Рунге – Кутты – Фельберга

Метод Рунге – Кутта – Фельберга позволяет оценивать погрешность решения без применения двойного пересчета. Формулы метода дают одновременно решения четвертого и пятого порядков точности. Разность этих решений служит оценкой погрешности более точного решения пятого порядка. Найденная оценка может использоваться для корректировки величины шага приращения аргумента. Метод, предложенный Фельбергом, является модификацией метода Рунге – Кутты:

$$y_{n+1} = y_n + h\left(\frac{25k_1}{216} + \frac{1408k_3}{2565} + \frac{2197k_4}{4104} - \frac{k_5}{5}\right),$$

$$x_{n+1} = x_n + h,$$

$$k_1 = f(x_n, y_n),$$

$$k_2 = f\left(x_n + \frac{h}{4}, y_n + \frac{hk_1}{2}\right),$$

$$k_3 = f\left(x_n + \frac{3h}{8}, y_n + \frac{3hk_1}{32} + \frac{9hk_2}{32}\right),$$

$$k_4 = f\left(x_n + \frac{12h}{13}, y_n + \frac{1932hk_1}{2197} - \frac{7200hk_2}{2197} + \frac{7296hk_3}{2197}\right),$$

$$k_5 = f\left(x_n + h, y_n + \frac{439hk_1}{216} - 8hk_2 + \frac{3680hk_3}{513} - \frac{845hk_4}{4104}\right),$$

$$k_6 = f\left(x_n + \frac{h}{2}, y_n - \frac{8hk_1}{27} + 2hk_2 - \frac{3544hk_3}{2565} + \frac{1859hk_4}{4104} - \frac{11hk_5}{40}\right).$$

Как может показаться из формул, количество вычислений правой части уравнения здесь равно 6 (по количеству коэффициентов k). Но преимущество данного метода в том, что шаг не является постоянным в отличие от методов Рунге – Кутты и может модифицироваться по следующему алгоритму:

1) если $|E| > \varepsilon$, то шаг уменьшается вдвое $h^2 = \frac{1}{2} \cdot h^1$;

2) если $|E| < \varepsilon$, то шаг удваивается $h^2 = 2h^1$;

3) если $\frac{\varepsilon}{32} < |E| < \varepsilon$, то шаг не меняется.

В этих формулах ε – заданная точность, а E рассчитывается по следующей формуле:

$$E = \frac{1}{360}k_1 - \frac{128}{4275}k_3 - \frac{2197}{75240}k_4 + \frac{1}{50}k_5 + \frac{2}{55}k_6.$$

Такая модификация метода является как его достоинством, так и его недостатком. Недостаток заключается в том, что количество шагов расчета заранее неизвестно, так как шаг является модифицируемым. Но в результате правильного выбора начальных значений количество вычислений правой части может быть значительно снижено по сравнению с классическим методом Рунге – Кутты, что и является достоинством этого метода.

Рассмотрим указанные методы для нескольких видов уравнений.

Уравнение вида $2xy(x) + y'(x) = 0$ при $y(1) = 2$.

Вычислим погрешности вычисления, занесенные в таблицу погрешностей (табл. 1), в зависимости от выбранного метода вычислений и погрешностей (табл. 1) и в зависимости от количества итераций (табл. 2), построим график (рис. 1), произведем анализ эффективности их использования.

Таблица 1. – Результаты расчетов погрешностей

Метод	Погрешность			
	$h = 0,01$	$h = 0,05$	$h = 0,1$	$h = 0,2$
Метод Эйлера	0,0000981	0,000399	0,000601	0,00671
Модифицированный метод Эйлера	0,00000158	0,0000469	0,000252	0,00267
Метод Рунге – Кутты третьего порядка	2,23E-08	0,00000326	0,0000313	0,000307
Метод Рунге – Кутты четвертого порядка	2,27E-10	0,000000168	0,00000333	0,0000858
Метод Рунге – Кутты – Фелберга	1,45E-08	1,45E-08	1,44E-08	1,43E-08

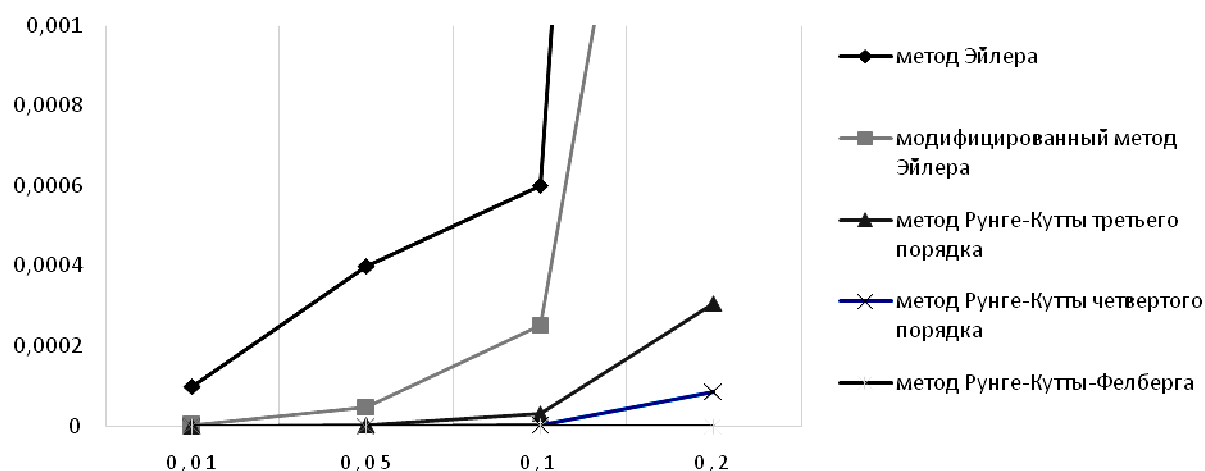


Рисунок 1. – Изменения погрешностей вычислений

Таблица 2. – Результат расчетов погрешностей в зависимости от количества итераций

Метод	Погрешность						
	5	10	20	40	55	56	100
Метод Эйлера		0,00671	0,000601	0,000399	–	–	–
Модифицированный метод Эйлера	0,00267	0,000252	0,0000469	–	–	–	0,00000158
Метод Рунге – Кутты третьего порядка	–	0,000307	0,0000313	3,26E–06	–	–	–
Метод Рунге – Кутты четвертого порядка	–	0,0000858	0,00000333	1,68E–07	–	–	–
Метод Рунге – Кутты – Фельберга	–	–	–	–	1,45E–08	1,45E–08	–

Из графиков и таблицы видно, что шаг значительно влияет на точность результатов первых четырех методов, а для метода Рунге – Кутты – Фельберга начальный шаг сказывается незначительно. При этом из таблицы хорошо видно, для достижения точности, близкой к точности метода Рунге – Кутты – Фельберга, в других методах может потребоваться гораздо большее количество итераций с небольшим постоянным шагом.

Заключение

Для решения дифференциальных уравнений в задачах строительной механики целесообразно применять методы Рунге – Кутты.

Метод 4-го порядка получает ответ с более высокой точностью по сравнению с методами более низкого порядка. Но отдельной задачей является поиск оптимального шага, так как при неудачном его выборе модифицированный метод Рунге – Кутты – Фельберга может оказаться значительно эффективнее.

ЛИТЕРАТУРА

1. Демидович, Б.П. Численные методы анализа / Б.П. Демидович, И.А. Марон, Э.З. Шувалова. – М. : Физматгиз, 1963. – 400 с.
2. Мак-Кракен, Д. Численные методы и программирование на ФОРТРАНе / Д. Мак-Кракен, У. Дорн. – М. : Мир, 1977. – 584 с.
3. Зельдович, Я.Б. Элементы прикладной математики / Я.Б. Зельдович, А.Д. Мышкис. – М. : Наука, 1972. – 592 с.
4. Хемминг, Р.В. Численные методы для научных работников и инженеров / Р.В. Хемминг. – М. : Наука, 1968. – 400 с.
5. Самарский, А.А. Численные методы / А.А. Самарский, А.В. Гулин. – М. : Наука, 1989. – 430 с.
6. Современные численные методы решения обыкновенных дифференциальных уравнений. – М. : Мир, 1979.
7. Бабушка, И. Численные процессы решения дифференциальных уравнений / И. Бабушка, Э. Витасек, М. Прагер. – М. : Мир, 1969. – 368 с.

УДК 519.6

ПРИМЕНЕНИЕ МЕТОДА РУНГЕ-КУТТЫ 4-ГО ПОРЯДКА ДЛЯ РЕШЕНИЯ ЗАДАЧ СТРОИТЕЛЬНОЙ МЕХАНИКИ

Ю.В. КОШЕЛЕВА, А.Н. ВОЛКОВ
(Представлено: Т.М. ГЛУХОВА)

Рассмотрены методы применения метода Рунге – Кутты 4-го порядка для решения задач строительной механики. Проанализированы особенности метода на простых примерах.

Одной из характерных особенностей научно-технического процесса является широкое применение численных математических методов и ЭВМ в различных областях творческой деятельности человека, и тем более в расчетах строительных конструкций.

При комплексном подходе к решению сложных задач строительной механики аналитические методы в большинстве случаев малоэффективны. Решение современных задач строительной механики связано с использованием сложных расчетных схем, близких к реальным конструкциям. Расчет таких сложных систем стал возможным только благодаря широкому применению численных методов расчета, ориентированных на использование ЭВМ.

В строительной механике задачи определения параметров напряженно-деформированного состояния сводятся к решению дифференциальных уравнений или системе дифференциальных уравнений, подобных следующему:

$$\begin{cases} \frac{d}{dx} s = \varepsilon_{s,ad}(\sigma_{s,ad}) - \varepsilon_{ct} \left(\frac{N - \sigma_{s,ad} \cdot A_{s,ad}}{A_{c,eff}} \right); \\ \frac{d}{dx} \sigma_{s,ad} = \frac{4}{\varnothing} \cdot \Omega_y \cdot \Omega_{p,tr} \cdot \tau(s), \end{cases} \quad (1)$$

где значения напряжений $\sigma_{s,ad,(l)}$ и $\sigma_{s,ad,(r)}$ и другие параметры известны, а значения смещений $s_{(l)}$ и $s_{(r)}$ неизвестны.

Для решения подобных задач применяют приближенные методы вычисления, так как применение прямых методов невозможно в силу сложности расчета. Среди численных методов, которые применяются для решения дифференциальных уравнений выделяют метод Рунге – Кутты 4 порядка:

Метод Рунге – Кутты 4-го порядка

Рассмотрим дифференциальное уравнение

$$y' = f(x, y); \quad (2)$$

с начальным условием

$$y = (x_0) = y_0. \quad (3)$$

Классический метод Рунге – Кутты 4-го порядка описывается следующей системой пяти равенств:

$$y_{i+1} = y_m + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad (4)$$

где

$$\begin{aligned} k_1 &= f(x_i, y_i), \\ k_2 &= f\left(x_i + \frac{h}{2}, y_i + \frac{hk_1}{2}\right), \\ k_3 &= f\left(x_i + \frac{h}{2}, y_i + \frac{hk_2}{2}\right), \\ k_4 &= f(x_i + h, y_i + hk_3). \end{aligned}$$

В данном случае мы имеем метод 4-го порядка, который является одним из наиболее применяемых на практике, так как обеспечивает высокую точность и в то же время отличается сравнительной простотой.

Свойства метода

1. Методы Рунге – Кутты являются одноступенчатыми: чтобы найти значение y_{m+1} , нужна информация о предыдущей точке x_m, y_m .
2. Методы Рунге – Кутты согласуются с рядом Тейлора вплоть до членов порядка h^p , где степень p различна для различных методов и называется порядковым номером или порядком метода.
3. Методы Рунге – Кутты не требуют вычисления производных от $f(x, y)$, а требуют вычисления самой функции.

Пример 1

Вычислить методом Рунге – Кутты интеграл дифференциального уравнения

$$y' = f(x, y)$$

при начальном условии $y(0) = 1$ на отрезке $[0, 0.5]$ с шагом интегрирования $h = 0.1$.

Решение. Вычислим y_1 .

Для этого сначала последовательно вычисляем k_j :

$$k_1 = x_0 + y_0,$$

$$k_1 = x_0 + y_0 = 0 + 1 = 1,$$

$$k_2 = x_0 + \frac{h}{2} + y_0 \frac{hk_1}{2} = (0 + 0.05) + (1 + 0.05) = 1.1,$$

$$k_3 = x_0 + \frac{h}{2} + y_0 \frac{hk_2}{2} = (0 + 0.05) + (1 + 0.005) = 1.105,$$

$$k_4 = x_0 + h + y_0 + hk_3 = (0 + 0.1) + (1 + 0.1105) = 1.2105.$$

Теперь получим

$$\Delta y_0 = \frac{0.1}{6} (1 + 2 \cdot 1.1 + 2 \cdot 1.105 + 1.2105) = 0.1103,$$

и, следовательно,

$$y_1 = y_0 + \Delta y_0 = 1 + 0.1103 = 1.1103.$$

Аналогично вычисляются последующие приближения.

Итак, $y(0.5) = 1.7974$.

Для сравнения точное решение дифференциального уравнения (1):

$$y = 2e^x - x - 1,$$

откуда

$$y(0.5) = 2\sqrt{e} - 0.5 - 1 = 1.79744.$$

Таким образом, точное и численное решения уравнения (1) совпали до пятого десятичного знака.

Если исследовать зависимость точности решения от шага, выбранного для расчета, то получим следующую закономерность, представленную на рисунке 1.

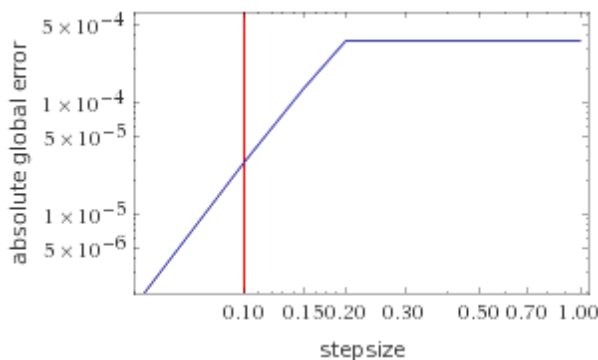


Рисунок 1. – Зависимость точности решения от шага

Рассмотрим другие виды уравнений.

Пример 2

$$y' = x \cdot y, y(0) = 2$$

Построим график зависимости точности от шага (рисунок 2).

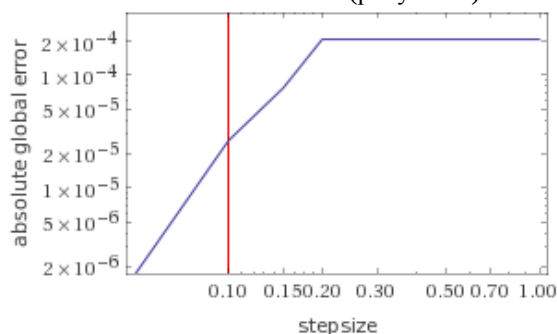


Рисунок 2. – График зависимости точности от шага

Пример 3

$$y' = -2 \cdot x \cdot y, y(0) = 2$$

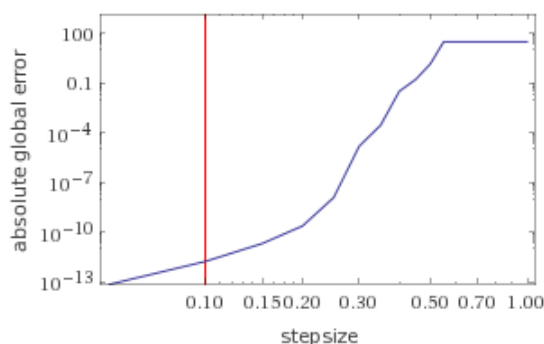


Рисунок 3. – График зависимости точности от шага

Из приведенных примеров можно сделать вывод, что величина шага, выбранная для решения дифференциального уравнения методом Рунге – Кутты, может существенно влиять на точность результатов. Но, начиная с определенного момента, значение шага практически не влияет на погрешность (график параллельно оси), а это значит, что благодаря большому шагу поиск решения займет меньше времени, но с потерей точности. Потому для применения метода Рунге – Кутты в задачах строительной механики требуется рассмотреть вопрос о выборе оптимального шага для поиска решения.

Исходя из вышеизложенного, для решения систем дифференциальных уравнений выбираем метод решения – метод Рунге – Кутта 4 порядка, обладающего рядом достоинств для решения задач строительной механики: метод является одноступенчатым; требует информацию только об одной точке; имеет небольшую погрешность; значение функции рассчитывается при каждом шаге.

Заклчение

Метод Рунге – Кутты четвертого порядка в работе детально рассмотрен. Приведены необходимые теоретические сведения, обоснована необходимость использования данного метода.

ЛИТЕРАТУРА

1. Демидович, Б.П. Численные методы анализа / Б.П. Демидович, И.А. Марон, Э.З. Шувалова. – М. : Физматгиз, 1963. – 400 с.
2. Мак-Кракен, Д. Численные методы и программирование на ФОРТРАНе / Д. Мак-Кракен, У. Дорн. – М. : Мир, 1977. – 584 с.
3. Зельдович, Я.Б. Элементы прикладной математики / Я.Б. Зельдович, А.Д. Мышкис. – М. : Наука, 1972. – 592 с.
4. Хемминг, Р.В. Численные методы для научных работников и инженеров / Р.В. Хемминг. – М. : Наука, 1968. – 400 с.
5. Самарский, А.А. Численные методы / А.А. Самарский, А.В. Гулин. – М. : Наука, 1989. – 430 с.
6. Современные численные методы решения обыкновенных дифференциальных уравнений. – М. : Мир, 1979.
7. Бабушка, И. Численные процессы решения дифференциальных уравнений / И. Бабушка, Э. Витасек, М. Прагер. – М. : Мир, 1969. – 368 с.

УДК 621.396.969

**ФОКУСИРОВКА РАДИОЛОКАЦИОННЫХ ИЗОБРАЖЕНИЙ
С ИСПОЛЬЗОВАНИЕМ ПАКЕТА MATLAB****С.А. ИГНАТЬЕВА***(Представлено: канд. техн. наук, доц. Р.П. БОГУШ)*

Представлена программная модель формирования РЛИ из необработанных радиолокационных данных дистанционного зондирования Земли, предоставляемых спутниками ERS-1 и ERS-2, с использованием MatLab. Приведены результаты практических исследований.

Радиолокационные системы используют облучение земной поверхности зондирующим сигналом, прием и обработку отраженных сигналов, и формирование выходного радиолокационного изображения (РЛИ) [1].

Процесс формирования РЛИ называется фокусировкой РГГ, выполняется на этапе первичной обработки радиолокационных данных и состоит из трех основных шагов: сжатие по дальности, компенсация миграции дальности и сжатие по азимуту. Процесс сжатия изображения по азимуту как раз и представляет собой синтез апертуры [2].

В настоящее время используются различные алгоритмы фокусировки РГГ:

- алгоритм быстрой свертки без компенсации миграции дальности;
 - алгоритм быстрой свертки с компенсацией миграции дальности (спектр-доплеровский алгоритм RDA – Range Doppler Algorithm); RDA с вторичным сжатием по дальности;
 - алгоритм гармонического анализа (SPECAN – Spectrum Analysis);
 - алгоритм ЛЧМ-масштабирования (CSA – Chirp Scaling Algorithm);
 - расширенный алгоритм ЛЧМ-масштабирования (ECS – Extended Chirp Scaling) и др.
- Также существует множество модификаций и дополнений к этим алгоритмам [1; 3; 4].

Преимуществом алгоритма быстрой свертки является высокая производительность, простота перестройки, получение выходного РЛИ без масштабных искажений при изменении наклонной дальности. Однако данный алгоритм нельзя применять при прожекторном режиме съемки и при нефокусированном синтезе, так как требуемая длина синтезированной апертуры оказывается много меньше ДНА, а азимутальное разрешение превышает количество отсчетов по азимуту [1; 2; 5].

При значительных углах отклонения от строго бокового обзора возникает нелинейность доплеровских сдвигов частоты, что приводит к дополнительной паразитной модуляции принимаемых импульсов, которая зависит от дальности и доплеровской частоты. Устранить эту модуляцию позволяет вторичное сжатие по дальности. Модификация алгоритма RDA дает возможность получения высококачественных РЛИ даже при достаточно больших отклонениях от строго-бокового обзора. Основным недостатком этого алгоритма является большое количество операций с плавающей точкой [2].

Алгоритм гармонического анализа имеет преимущества в скорости обработки, если используется для синтеза РЛИ в случае, когда РГГ содержит участки, в которых съемка не велась, или если необходимо получить обзорное изображение с низким разрешением. В противном случае, алгоритм теряет свои преимущества в скорости обработки. В настоящее время используется редко, из-за необходимости перескритизации сигнала в столбцах РГГ [1; 2].

Несмотря на разнообразие алгоритмов и их модификаций для формирования РЛИ, каждый из них имеет ряд ограничений при использовании, направленных на необходимость поиска компромисса между точностью полученных РЛИ и вычислительной сложностью. Алгоритм RDA обладает высокой производительностью, позволяет получать РЛИ без масштабных искажений, учитывает зависимость наклонной дальности от центральной доплеровской частоты, может использоваться при отклонениях от строго бокового обзора.

Для формирования радиолокационного изображения с использованием представленных в формате CEOS необработанных данных ДЗЗ на основе алгоритма быстрой свертки с компенсацией миграции дальности необходимо решить следующие задачи:

- считать из файла заголовка метаданные, а из файла с данными получить квадратурные и синфазные значения сигнала;
- рассчитать параметры, необходимые для формирования опорных функций по дальности и по азимуту;
- рассчитать частотный сдвиг для каждого пикселя изображения для компенсации миграции дальности;
- выполнить процедуры сжатия по дальности и по азимуту.

При моделировании алгоритма формирования РЛИ реализованы и использовались следующие функции:

Ref_r – функция рассчитывает период дискретизации быстрого времени, рассчитывает количество отсчетов и формирует массив значений, принимаемых быстрым временем. Рассчитывает фазу переданного сигнала по дальности и формирует опорную функцию по дальности. Опорная функция по дальности рассчитывается только один раз (на первой итерации) и в дальнейшем не меняется (Листинг 1).

Листинг 1 – MatLab-функция для формирования опорной функции по дальности

```
function chirp=ref_r(fc,slope,tau,fs,nValid);
% fc – центральная частота; slope – угол отклонения ЛЧМ (оценка ЛЧМ); tau – длительность %импульса;
% fs – частота дискретизации; nValid – количество отсчетов
dt = 1/fs; % Шаг изменения времени
npts = floor(tau*fs); % Количество отсчетов времени
t = [-npts/2:npts/2-1]*dt; % Массив значений времени
phase = 2*pi*fc*t+pi*slope*t.^2; % фаза переданного сигнала
% Построение графика опорной функции по дальности:
figure
chirp = exp(i*phase);
plot(t, chirp)
title('ЛЧМ-сигнал')
chirp=[exp(i*phase) zeros(1,nValid,length(phase))]
```

Comp_r – функция выполняет процедуру сжатия данных по дальности. Для этого на каждой итерации производится операция БПФ, свертка с комплексно-сопряженной опорной функцией, и операция ОБПФ (Листинг 2).

Листинг 2 – MatLab-функция для сжатия по дальности

```
function data=comp_r(ref,data,nValid,numLines)
% ref – Опорная функция по дальности; data – Необработанные данные; nValid – номер отсчета %
% по дальности; numLines – номер строки по азимуту
fftlen = 2^ceil(log2(nValid)); % ближайшее большее целое, равное степени 2
fftref = conj(fft(ref,fftlen)); % комплексно-сопряженная функция
% для БПФ, вычисленной для fftlen
% точек для опорной функции БПФ для % (2048
8192)

data = fft(data,fftlen,2);
data = ifft(data .* repmat(fftref,numLines,1,[],2));
data(:,nValid+1:end) = [];
```

Dop_centр_fr – функция используется для определения центральной доплеровской частоты и использует метод среднего изменения фазы. Рассчитывается фаза каждого отраженного сигнала, находится разность между соседними отсчетами по азимуту для каждого значения по дальности. Затем находится среднее арифметическое значение всех средних значений изменения фаз между соседними отсчетами.

Chirp_az – рассчитывает фазу переданного сигнала по азимуту и формирует опорную функцию по азимуту.

Ref_az – функция используется для формирования опорной функции по азимуту для каждого отсчета по дальности (Листинг 3). Определяет наклонную дальность как функцию доплеровской частоты, ширину раскрыва антенны на уровне 3 дБ, создает массив значений медленного времени, определяет количество действительных отсчетов по азимуту (таким образом, вычисляется, насколько каждая соседняя часть РЛИ будет перекрывать соседнюю часть) и генерируется массив значений опорной функции для каждого отсчета по дальности. Использует для этого функцию Chirp_az.

Листинг 3 – MatLab-функция для формирования опорной функции по азимуту для каждого отсчета по дальности

```
function [az_ref,validAzPts] = ref_az(doppler,nValid,fs,ro,vEff,lambda,prf,numLines)
% doppler : центральная доплеровская частота; nValid : номер отсчета по дальности; fs : частота % дискретизации; ro : ближняя наклонная дальность; vEff : путевая скорость платформы; lambda : длина % волны; prf : Частота следования импульсов; numLines : номер отсчета по азимуту
L=10;
c= 2.99792458e+08;
% Вычисление дальности перпендикулярно направлению движения:
range = ro + [0:nValid-1] * (c/(2*fs));
rdc = range/sqrt(1-(lambda*doppler/(2*vEff))^2);           % Наклонная дальность
fRate = -(2*vEff^2/lambda)./rdc;                          % Оценка доплеровской частоты
az_beamwidth = rdc * (lambda/L) * 0.8;                   % ширина раскрытия антенны на уровне 3дБ
tau_az = az_beamwidth / vEff;                             % массив времени, в течение % которого точка находится в пределах ДНА
nPtsAz = ceil(tau_az(end) * prf);                         % Используется для дальней дальности
validAzPts = numLines - nPtsAz ;                          % количество действительных % отсчетов по азимуту (размер апертуры)

% Генерация опорной функции по азимуту для каждого отсчета по дальности
for rangeBin=1:nValid
    az_ref(:,rangeBin) = chirp_az(doppler, fRate(rangeBin),tau_az(rangeBin), prf, numLines);
end
```

R_migr – функция, используемая для компенсации миграции дальности. Определяет величину частотного сдвига для каждого отсчета по азимуту, прибавляет к соответствующим отсчетам величину смещения.

Comp_az – функция выполняет процедуру сжатия по азимуту. Для этого выполняется свертка с опорной функцией по азимуту данных, полученных после компенсации миграции дальности, и процедура ОБПФ.

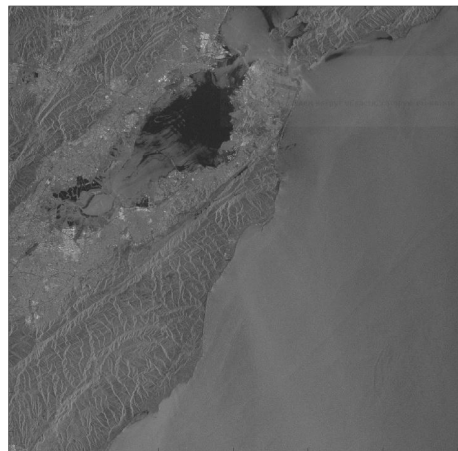
Для формирования РЛИ из необработанных данных, предоставляемых спутниками ERS-1 и ERS-2, используются следующие характеристики: $L = 10$ м, $h = 782\,000$ м, $K_r = 4,9 \cdot 10^{11}$ [6; 7].

Спутником ERS-2 часть метаданных, необходимых для формирования РЛИ уровня 1А, не предоставляется, в числе которых длительность строка дальности и углы падения к ближней, центральной и дальней точкам.

При моделировании алгоритма формирования РЛИ это учитывается при расчете ближней наклонной дальности. Результаты фокусировки для данных, предоставляемых спутниками ERS-1, показаны на рисунке 1а, ERS-2 – на рисунке 1б.



а)



б)

Рисунок 1. – Сфокусированные изображения с геометрической коррекцией

Заключение

Проведенное исследование позволило получить следующие результаты:

- разработана программная реализация модели формирования РЛИ из необработанных радиолокационных данных дистанционного зондирования Земли, предоставляемых спутниками ERS-1 и ERS-2, с использованием MatLab;
- проанализированы и учтены особенности предоставленных данных;
- реализована возможность анализа промежуточных результатов при формировании РЛИ на различных этапах обработки.

ЛИТЕРАТУРА

1. Радиолокационные системы землеобзора космического базирования / В.С. Верба [и др.]. – М. : Радиотехника, 2010. – 680 с.
2. Ушенкин, В.А. Анализ алгоритмов фокусировки радиоголограмм от космического радиолокатора с синтезированной апертурой антенны // Информатика и прикладная математика : сб. науч. тр. / В.А. Ушенкин ; Рязан. гос. ун-т имени С.А. Есенина (Рязань). – 2014. – № 20. – С. 95–100.
3. ESA earthnet online [Electronic resource]. – Mode of access: <https://earth.esa.int/handbooks/asar/CNTR2-6-1-2-3.html>. – Date of access: 12.04.2016.
4. Processing the Azimuth –Variant Bistatic SAR Data by Using Monostatic Imaging Algorithms Based on Two-Dimensional Principle of Stationary Phase [Electronic resource]. – Mode of access: https://www.researchgate.net/publication/224230689_Processing_the_Azimuth-Variant_Bistatic_SAR_Data_by_Using_Monostatic_Imaging_Algorithms_Based_on_Two-Dimensional_Principle_of_Stationary_Phase. – Date of access: 19.01.2017.
5. Бакулев, П.А. Радиолокационные системы : учеб. для вузов / П.А. Бакулев. – М. : Радиотехника, 2004. – 320 с.
6. ESA. Earth Online [Electronic resource]. – Mode of access: <https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/ers/satellite>. – Date of access: 30.12.2016.
7. ESA. Earth Online. [Electronic resource]. – Mode of access: https://earth.esa.int/web/guest/missions/esa-operational-eomissions/ers/satellite#_56_INSTANCE_uBh2_matmp. – Date of access: 30.12.2016.

УДК 628.972

РАСЧЕТ ОСВЕЩЕННОСТИ В ПРОГРАММНОМ ПАКЕТЕ DIALUX

Е.А. РОДЧЕНКОВА

(Представлено: канд. техн. наук, доц. А.Л. АДАМОВИЧ)

Проведен расчет освещенности в аудитории при помощи пакета прикладных программ DIALux. Получены значения освещенности в контрольных точках. Выполнены экспериментальные измерения освещенности. Представлены результаты сравнения расчетной и измеренной освещенности.

Цель данной работы – сравнение распределения освещенности полученного в программе DIALux с результатами экспериментальных измерений освещенности в помещении.

В качестве помещения была выбрана аудитория УО «ПГУ» длиной 9,28 м, шириной 5,6 м и высотой потолка 4 м, со светлыми потолком и стенами и темным полом и нормальной окружающей средой. В соответствии с этим были выбраны коэффициенты отражения потолка, стен и пола 0,7; 0,6 и 0,3 соответственно, коэффициент запаса $K_z = 1,4$ [1]. В данной аудитории размещено 12 потолочных одноламповых светильников ЛПП-1х58 с трубчатыми люминесцентными лампами мощностью 58 Вт каждая (как показано на рисунке 1).

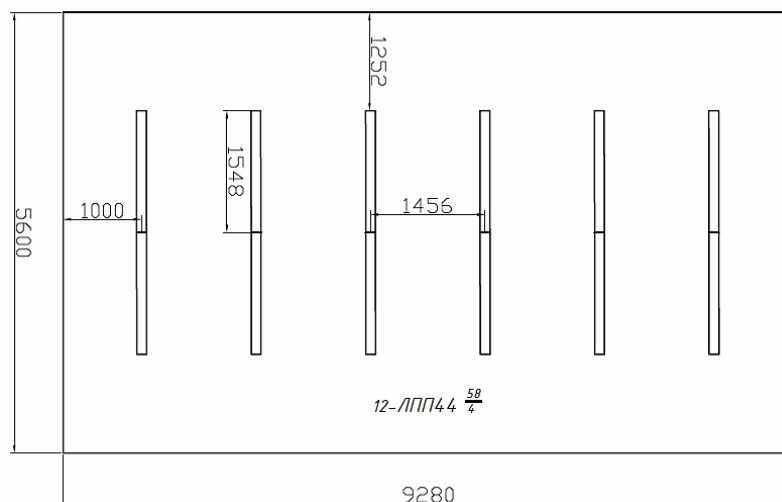


Рисунок 1. – План потолка в аудитории

Расчет освещенности в программе DIALux

Ход выполнения:

1. В окне «Редактор помещения» задаем размеры помещения. Во вкладке «Поверхности» в помещении вводим коэффициенты отражения поверхностей.
2. Во вкладке «Метод плана технического обслуживания» задаем коэффициент уменьшения равный $1/K_z = 0,7$. Далее в соответствующей вкладке задаем высоту рабочей поверхности 0,8 м, для которой будет определяться освещенность.
3. Из базы светильников предварительно выбираем светильник ЛПП44-58-012 производства ОАО «Ардатовский светотехнический завод». На панели инструментов выбирается «Отдельный светильник» и размещается на необходимое место в соответствии с планом (рис. 1). Аналогично размещаются все остальные светильники.
4. Выполняется расчет.

Экспериментальное измерение освещенности

Для измерения освещенности использовался люксметр типа ТКА-ПКМ-0.5, имеющий диапазон измерений освещенности 1,0–200 000 лк и предел допускаемой основной относительной погрешности измерения освещенности не более $\pm 8,0\%$.

Измерения произведены по методике [2] в вечернее время в отсутствие естественного освещения при плотно завешенных окнах.

Контрольные точки располагались на высоте 0,8 м от пола: 3 точки вдоль рядов и 3 точки между рядами светильников. Результаты измерения и расчета представлены в таблице.

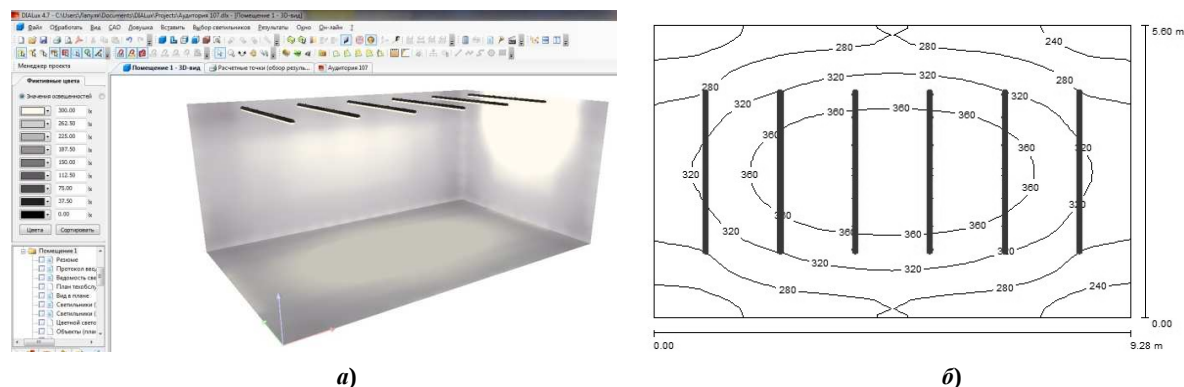


Рисунок 2. – Распределение освещенности в цехе:
3D модель (а) и на расчетной плоскости (б)

Таблица. – Результаты измерений и расчета освещенностей, лк, в контрольных точках (рассчитано/измерено)

Номер ряда	Рассчитано			Измерено		
	Точка 1	Точка 2	Точка 3	Точка 1	Точка 2	Точка 3
Ряд 1	288	334	288	295	342	313
Между рядами	288	377	310	309	397	321
Ряд 2	324	310	377	344	316	400
Между рядами	324	334	389	343	342	403
Ряд 3	335	339	395	356	348	404
Между рядами	340	341	397	364	351	416
Ряд 4	342	339	395	365	356	410
Между рядами	340	334	389	349	348	416
Ряд 5	335	324	377	363	332	382
Между рядами	324	309	360	347	322	373
Ряд 6	310	287	334	322	303	359

Согласно результатам отличие рассчитанных и измеренных значений освещенностей оказывается небольшим: не более 10% на центральных и не более 20% на крайних контрольных точках аудитории. В соответствии с ТКП 45-2.04-153-2009 (02250) «Естественное и искусственное освещение. Строительные нормы проектирования» [1] освещенность рабочих поверхностей на уровне 0,8 м от пола для аудиторий высших учебных составляет 400 лк при общем искусственном освещении.

Заключение

Расчет освещенности в DIALux с большой степенью достоверности отражает реальные значения освещенности в проектируемых помещениях. Кроме того DIALux позволяет экономить время проектировщика, учитывать расстановку оборудования в помещении, одновременное действие искусственного и естественного освещения, имеет базу данных светильников большинства производителей, а результаты такого автоматизированного расчета обладают наглядностью.

ЛИТЕРАТУРА

1. Естественное и искусственное освещение. Строительные нормы проектирования : ТКП 45-2.04-153-2009 (02250). – Введ. 2010-01-01. – Минск : Минстройархитектуры, 2010. – 110 с.
2. Здания и сооружения. Методы измерения освещенности : ГОСТ 24940-96. – Введ. 1997-01-01. – М. : Минстрой России, ГУП ЦПП, 1997. – 24 с.

УДК 628.972

**СВЕТОТЕХНИЧЕСКИЙ РАСЧЕТ ОСВЕЩЕНИЯ ЦЕХА
В ПРОГРАММНОМ ПАКЕТЕ DIALUX****Е.А. РОДЧЕНКОВА***(Представлено: канд. техн. наук, доц. А.Л. АДАМОВИЧ)*

Приведено краткое описание и возможности программного пакета DIALux. Выполнены светотехнические расчеты освещения цеха аналитическим методом и с применением программного пакета DIALux. Представлены результаты расчетов и проведено их сравнение.

DIALux – программа для планирования и дизайна освещения, разрабатываемая с 1994 года DIAL GmbH – Немецким институтом прикладной светотехники. В настоящее время программа DIALux позиционируется как универсальный программный комплекс для проведения светотехнических расчетов искусственного освещения как внутреннего, так и наружного. При расчете в программе DIALux используются встроенные базы данных светильников от мировых производителей, которые постоянно обновляются и доступны для свободной загрузки в сети Интернет.

Цель данной работы – сравнение результатов аналитического и компьютерного расчета в программе DIALux светотехнического расчета общего рабочего освещения цеха. В результате светотехнических расчетов определяется мощность ламп в светильниках и их количество, а также размещение светильников на плане потолка для создания освещенности на рабочих поверхностях не ниже нормируемой.

Исходные данные: механический цех имеет длину $A = 48$ м, ширину $B = 30$ м и высоту потолка $H = 7,7$ м; коэффициенты отражения потолка, стен и пола 0,5; 0,3 и 0,1 соответственно, нормируемая освещенность 300 лк и коэффициент запаса 1,4 выбраны из [1], окружающая среда – нормальная, минимально допустимая степень защиты светильников в помещении – IP20.

Аналитический светотехнический расчет

При проведении данного расчета пользуемся методикой, изложенной в [2]. Предварительно принимаем КПД светильника $\eta = 70$ %, тип кривой силы света – «Д».

Определяем расчетную высоту подвеса светильника:

$$H_p = H - h_c - h_p = 7,7 - 1,5 - 0,8 = 5,4 \text{ м},$$

где h_c – расстояние от светильника до перекрытия (свес), м (принимается в диапазоне 0–1,5 м); h_p – высота расчетной поверхности над полом, м (принимается 0,8 м).

По таблице 7.1 [2, с. 104] по типу КСС «Д» светильника выбираем отношение $L / H_p = 1,5$.

Тогда наилучшее расстояние между соседними рядами светильников:

$$L = H_p \cdot 1,5 = 5,4 \cdot 1,5 = 8,1 \text{ м}.$$

Расстояния от стен до первого ряда:

$$l = 0,5 \cdot L = 0,5 \cdot 8,1 = 4,05 \text{ м}.$$

Число рядов светильников:

$$R = \frac{B - 2 \cdot l}{L} = \frac{30 - 2 \cdot 4,05}{8,1} = 2,7 \approx 3.$$

Число светильников в ряду:

$$N_R = \frac{A - 2 \cdot l}{L} = \frac{48 - 2 \cdot 4,05}{8,1} = 4,93 \approx 5.$$

Реальное расстояние между рядами:

$$L_B = \frac{B - 2 \cdot l}{R - 1} = \frac{30 - 2 \cdot 4,05}{3 - 1} = 11 \text{ м}.$$

Реальное расстояние между центрами светильников:

$$L_A = \frac{A - 2 \cdot l}{N_R - 1} = \frac{48 - 2 \cdot 4,05}{5 - 1} = 10 \text{ м}.$$

Для прямоугольных помещений проверяем условие:

$$1 \leq L_A / L_B \leq 1,5.$$

Так как $L_A / L_B = 10 / 11 = 0,9 < 1$ и условия не соблюдается, то согласно [2] увеличиваем число рядов на один. Принимаем $R = 4$.

$$L_B = \frac{B - 2 \cdot l}{R - 1} = \frac{30 - 2 \cdot 4,05}{4 - 1} = 7,3 \text{ м}.$$

$$1 \leq \frac{L_A}{L_B} = \frac{10}{7,3} = 1,37 \leq 1,5.$$

Общее число светильников:

$$N_{св} = R \cdot N_R = 4 \cdot 5 = 20 \text{ шт}.$$

Расчетное значение светового потока одной лампы в каждом светильнике:

$$\Phi_{лр} = \frac{E_n \cdot K_z \cdot F \cdot z}{N \cdot \eta_{oy}} = \frac{300 \cdot 1,4 \cdot 30 \cdot 48 \cdot 1,15}{20 \cdot 0,67} = 51904 \text{ лм},$$

где E_n – нормируемое значение освещенности, лк; K_z – коэффициент запаса; F – освещаемая площадь, м²; z – отношение средней освещенности к минимальной и для ламп типа ДРЛ $z = 1,15$; η_{oy} – коэффициент использования светового потока осветительной установки, о.е.

Определяем индекс помещения:

$$i_n = \frac{A \cdot B}{H_p \cdot (A + B)} = \frac{30 \cdot 48}{5,4 \cdot (30 + 48)} = 3,4.$$

По таблице 8.1 [2, с. 108] выбираем $\eta_{oy} = 0,67$ в зависимости от индекса помещения и коэффициентов отражения поверхностей.

По таблице 5.8 [2, с. 54] по полученному световому потоку выбираем лампы типа ДРЛ1000(6)-3 с ближайшим световым потоком $\Phi_L = 58000 \text{ лм}$.

Отклонение светового потока лампы от расчетного:

$$\frac{\Phi_L - \Phi_{лр}}{\Phi_{лр}} \cdot 100\% = \frac{58000 - 51904}{51904} = +11,7 \%.$$

Таким образом, определено, что для освещения заданного механического цеха необходимо разместить 20 светильников марки РСП05-1000 с лампами ДРЛ1000 согласно плану, приведенному на рисунке 1.

Светотехнический расчет в программе DIALux

Порядок выполнения расчета:

1. В окне «Редактор помещения» задаем размеры помещения. Во вкладке «Поверхности» в помещении вводим коэффициенты отражения поверхностей.

2. Во вкладке «Метод плана технического обслуживания» задаем коэффициент уменьшения равный $1/K_z = 0,7$. Далее в соответствующей вкладке задаем высоту рабочей поверхности 0,8 м, для которой будет определяться освещенность.

3. Из базы светильников предварительно выбираем светильник РСП05-1000-032 производства ОАО «Ардатовский светотехнический завод». Во вкладке «Монтаж» задаем нормированное значение освещенности 300 лк и величину свеса.

4. После нажатия кнопки «Предложить» происходит автоматическое размещение необходимого количества светильников по площади помещения, как показано на рисунке 2.

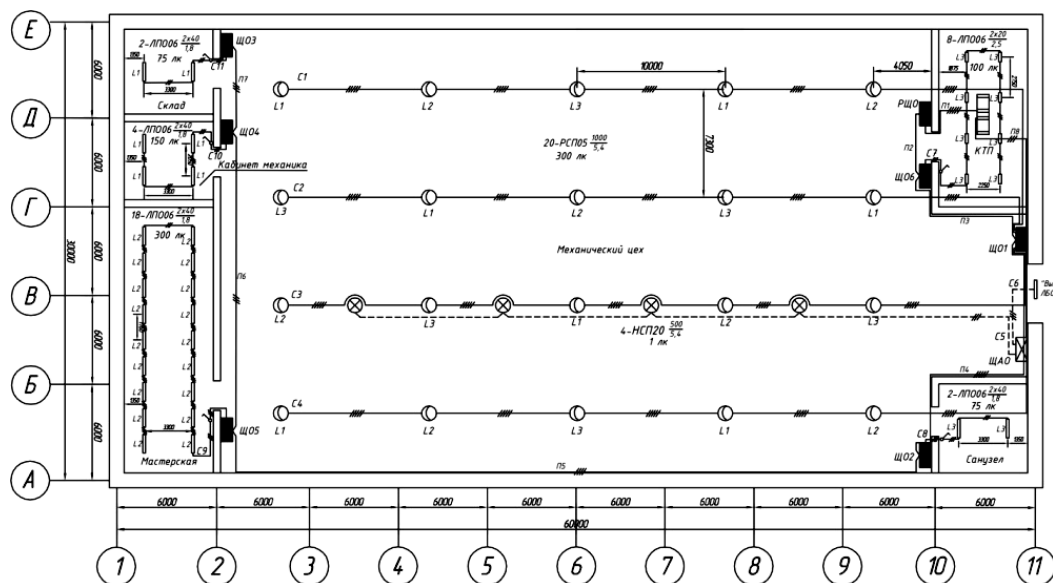


Рисунок 1. – План размещения светильников в цехе

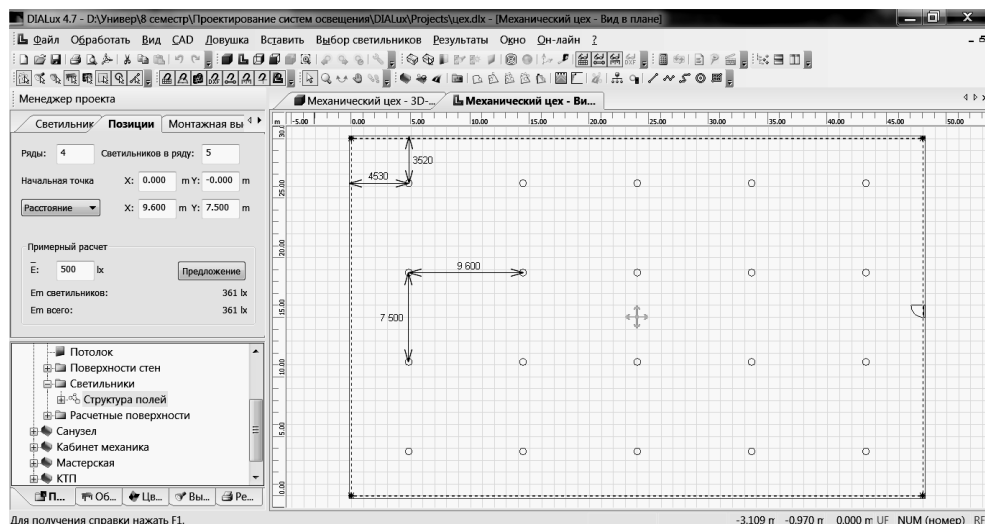
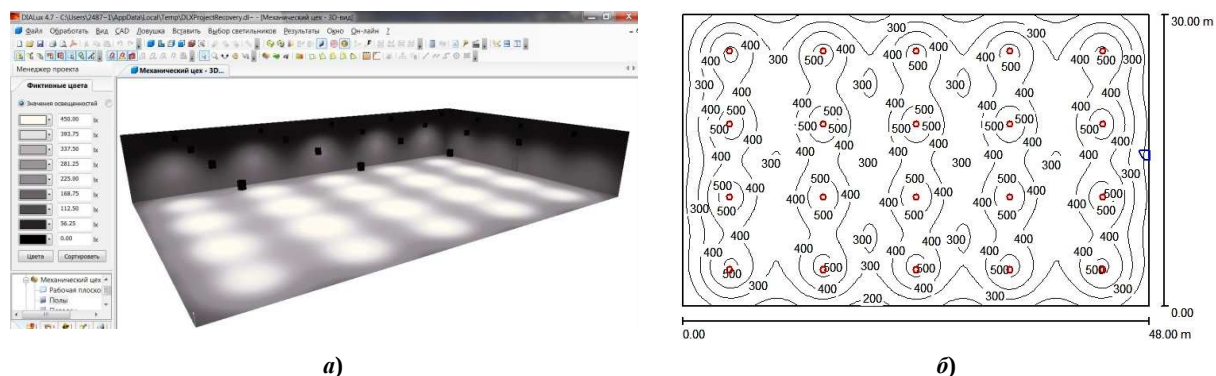


Рисунок 2. – План размещения светильников

Программа DIALux позволяет визуализировать значения освещенности, как показано на рисунке 3, что позволяет проектировщику оценить степень равномерности освещения и сделать необходимые корректировки.

Рисунок 3. – Распределение освещенности в цехе
на рабочей поверхности: 3D модель (а) и на плоскости (б)

Сравнение результатов светотехнических расчетов приведено в сводной таблице.

Таблица. – Результаты расчета

Параметр	Аналитический расчет	Расчет в DIALux
Тип светильников	РСП05-1000	РСП05-1000-032
Количество светильников	20	20
Количество рядов	4	4
Количество светильников в ряду	5	5
Расстояние между рядами	7,3 м	7,5 м
Расстояние между светильниками в ряду	10 м	9,6 м

Как видно, результаты светотехнического расчета двумя способами практически совпадают.

Заключение

Аналитический метод расчета внутреннего освещения хорошо разработан и дает достаточно точный результат. Данный метод рекомендован к использованию действующими стандартами.

Применение программного пакета DIALux для выполнения светотехнических расчетов позволяет проектировщику не только ускорить расчет по сравнению с аналитическим методом, но и произвести визуальную оценку выбранного варианта светотехнического решения.

ЛИТЕРАТУРА

1. Естественное и искусственное освещение. Строительные нормы проектирования : ТКП 45-2.04-153-2009 (02250). – Введ. 2010-01-01. – Минск : Минстройархитектуры, 2010. – 110 с.
2. Козловская, В.Б. Электрическое освещение : справочник / В.Б. Козловская, В.Н. Радкевич, В.Н. Сацукевич. – 2-е изд. – Минск : Техноперспектива, 2008. – 271 с.

УДК 62-1/-9

**СИСТЕМА ЭКСТРЕННОГО ОПОВЕЩЕНИЯ
ДЛЯ ЛЮДЕЙ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ****Н.Д. ЧУПРОВ****(Представлено: С.Н. АБРАМЕНКО)**

Описывается проект создания и разработки устройства беспроводного оповещения для людей с ограниченными возможностями. Представлено описание разработки структуры по требованиям служб МЧС и процесс создания макетного образца.

Необходимость контроля состояния людей с ограниченными возможностями – задача, не теряющая актуальности. Таким людям необходимы не только уход и внимание, но иногда и необходимость сообщить об экстренных ситуациях как своего состояния, так и происходящего в помещении. Не исключаются аварийные ситуации – пожар и задымление, резкое ухудшение состояния, посторонние запахи. Основная проблема – невозможность постоянного нахождения с такими людьми 24 часа в сутки, а зачастую и неспособность самих граждан адекватно и быстро отреагировать на опасные ситуации.

Выбор структуры устройства, постановка задачи

Среди систем оповещения в последнее время лидируют беспроводные. Особенно выгодны они стали с повсеместным распространением мобильной связи. Цифровые каналы мобильной связи в состоянии не только предавать речевые сообщения, но короткие sms и прочие виды информационных сообщений. Это позволит создавать систему оповещения с передачей как автоматической информации без участия человека, так и сигнал по его решению.

Приведем примерный перечень параметров для таких устройств:

- использования стандарта GSM для передачи сообщения органам опеки и безопасности (например, МЧС и скорой помощи);
- наличие резервного источника энергии на случай отказа сетевого электроснабжения;
- способность измерять и реагировать на несколько видов параметров окружающей среды:
 - задымление;
 - газ;
- принудительный вызов человека;
- способность оповещения нескольких служб по нескольким номерам;
- передача информации о типе вызова (какой датчик сработал) и адресе местонахождения посредством SMS;
- наличие дистанционного пульта в зоне доступности человека.

Обзор аналогов

Приведенные ниже модели оснащены беспроводной тревожной кнопкой с функцией отправки смс на несколько телефонных номеров (табл. 1).

Таблица 1. – Обзор и стоимость аналогов устройства

Аналог 1 [1] LPSECURITY 007	Аналог 2 [2]: Медицинская сигнализация A10	Аналог 3 [3]: Медицинская сигнализация «ML-100»
		
Стоимость от 135 до 421 BYN	Стоимость 219,86 BYN	Стоимость 160,67 BYN

Данная разработка, должна обеспечить конкурентоспособность как по цене, так и функциональности, прежде всего по наличию встроенных датчиков.

Структурная схема

На основе анализа задачи и функциональных аналогов была разработана структурная схема, приведенная на рисунке 1.

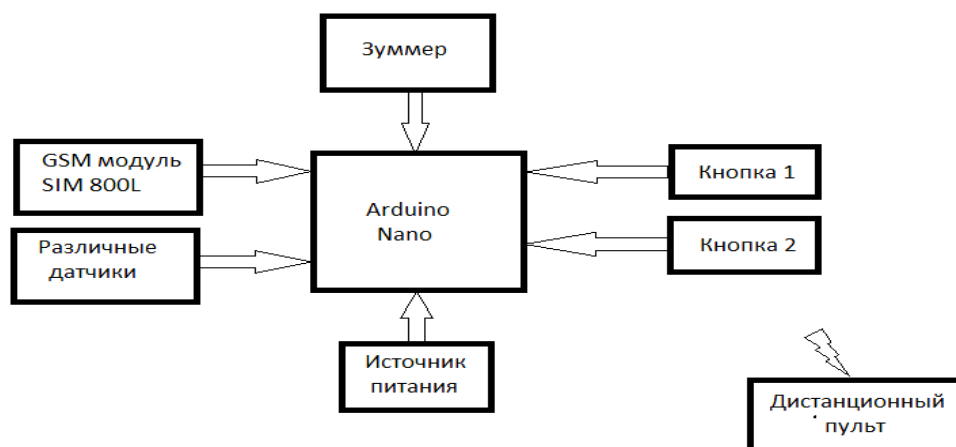


Рисунок 1. – Структурная схема

Состав устройства и датчики

Произведен обзор, и на основании цен и технических характеристик выбраны следующие составные части:

- аппаратная платформа Arduino Nano [4];
- GSM модуль SIM800L [4];
- зуммер – звукоизлучатель DIY Kit [4];
- беспроводная кнопка-брелок ALLOYSEED89360.02 [4];
- датчик угарного газа MQ-7 [4];
- датчик дыма MQ-2 [4];
- датчик пламени HWA YEH [4];
- резервный аккумулятор.

Пример реализации

На данных комплектующих собран макетный образец для отладки программного обеспечения, приведенный на рисунке 2.

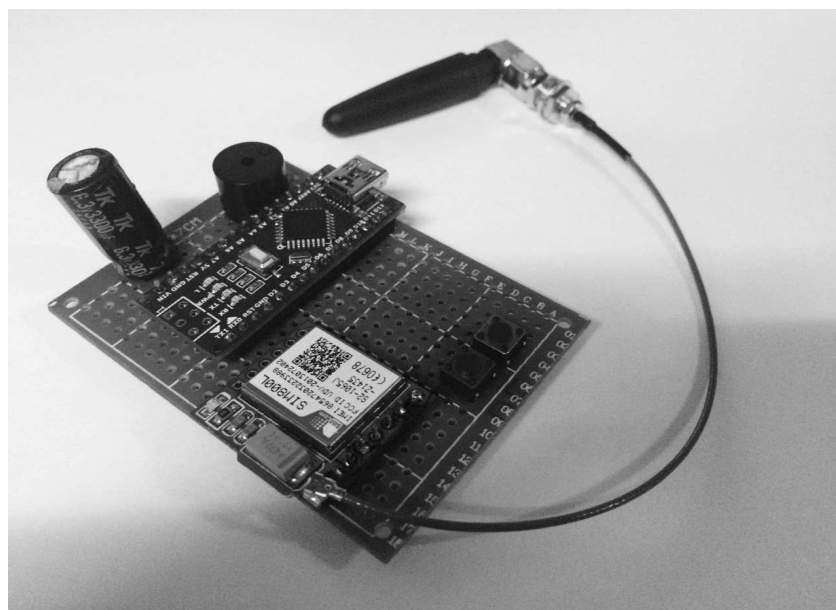


Рисунок 2. – Макетный образец

Описание работы

Система экстренного оповещения предназначена для экстренного вызова помощи. Устройство специально разработано для использования пожилыми людьми и людьми с ограниченными возможностями. Для начала работы с устройством необходимо установить в него SIM-карту любого оператора. В случае тревоги или резкого ухудшения самочувствия человеку необходимо нажать на кнопку на устройстве или на беспроводную кнопку SOS. Устройство сразу начинает отправлять SMS сообщения на запрограммированные номера. Отправка сообщения сопровождается звуковым сигналом, для оповещения рядом находящихся людей. Система имеет кнопку отмены, при не намеренном нажатии, человек может сообщить о ложном вызове.

Благодаря беспроводной кнопке пользователь может беспрепятственно передвигаться по квартире и в любой момент нажать тревожную кнопку.

Также устройство имеет возможность подключения различных датчиков, что позволяет обеспечить безопасность на более высоком уровне.

Заключение

Реализованное устройство уже доказало свою работоспособность на конкурсе с участием сотрудников МЧС и студентов.

Среди перспективных задач по модернизации следует отметить:

- возможность изготовления корпуса (например, с помощью 3D-печати);
- проработки вопроса с мобильными операторами со стороны МЧС на соответствующие тарифы.

ЛИТЕРАТУРА

1. Lpsecurity Беспроводной GSM SMS SOS [Электронный ресурс]. – 2013. – Режим доступа: https://ru.aliexpress.com/item/Quad-Band-Wireless-GSM-SMS-SOS-Emergency-Auto-Dialer-Alarm-System-with-1-Panic-Button-elderly/32672544412.html?spm=a2g0v.search0304.4.9.VJDIfb&aff_platform=aaf&cpt=1506531072664&sk=zj6qB6AIM&aff_trace_key=0033789819a6444f9bdb2ae55203b448-1506531072664-02253-zj6qB6AIM&terminal_id=95ea0483f11f497591c131d0ac793abb. – Дата доступа: 08.09.2017.
2. Медицинская сигнализация A10 [Электронный ресурс]. – 2014. – Режим доступа: <https://tiu.ru/p12597708-meditsinskaya-signalizatsiya-a10;all.html>. – Дата доступа: 09.09.2017.
3. Медицинская сигнализация [Электронный ресурс]. – 2015. – Режим доступа: <https://tiu.ru/p12598207-meditsinskaya-signalizatsiya-100;all.html>. – Дата доступа: 08.09.2017.
4. Интернет магазин Aliexpress [Электронный ресурс]. – 2012. – Режим доступа: <https://ru.aliexpress.com/?spm=a2g0v.search0104.1000002.1.XgdgWx>. – Дата доступа: 09.09.2017.
5. Основы сотовой связи стандарта GSM [Электронный ресурс]. – 2012. – Режим доступа: <http://www.kodges.ru/nauka/obrazovanie/78007-osnovy-sotovoj-svyazi-standarta-gsm.html>. – Дата доступа: 09.09.2017.
6. Баранов, В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / В.Н. Баранов. – М. : Издат. дом «Додэка-XXI», 2006. – 288 с.
7. Климова, Л.М. Основы практического программирования на языке Си / Л.М. Климова. – М. : Изд-во ПРИОР, 1999. – 464 с.

УДК 37.014.544.4

**РАЗРАБОТКА ЛАБОРАТОРНОГО СТЕНДА ПО ИЗУЧЕНИЮ ОСНОВ АВТОМТИЗАЦИИ
СТУДЕНТОВ СПЕЦИАЛЬНОСТЕЙ 43 01 03 ЭЛЕКТРОСНАБЖЕНИЕ
И 39 02 02 ПРОЕКТИРОВАНИЕ И ПРОИЗВОДСТВО ПРОГРАММНО-УПРАВЛЯЕМЫХ РЭС**

В.Ю. БОРИСЕНКО
(Представлено: С.Н. АБРАМЕНКО)

Описывается проект создания и разработки лабораторно-отладочного стенда для изучения студентами дисциплин «Монтаж, наладка и эксплуатация электрооборудования» и «Техническая эксплуатация и ремонт средств КИП и А».

Разработка стенда необходима для следующих целей:

- 1) изучение студентами правил монтажа энергетического образца;
- 2) знакомство с особенностями программирования современных средств автоматизации;
- 3) создание отладочного стенда, для разработки программ обеспечения программируемого логического контроллера (ПЛК).

Разрабатываемый стенд позволит производить демонстрации произведенных настроек и написанных программ с целью выявления и устранения ошибок данных систем.

Состав стенда

Стенд будет монтирован в стандартный ящик ЩМП-06-02, в котором будут размещены следующие функциональные блоки:

- автоматы защиты и коммутации;
- источник питания 24 В;
- частотный преобразователь;
- коммутационная плата;
- драйвер шагового двигателя;
- шаговый двигатель и асинхронный двигатель.

Структурная схема разрабатываемого стенда приведена на рисунке 1.

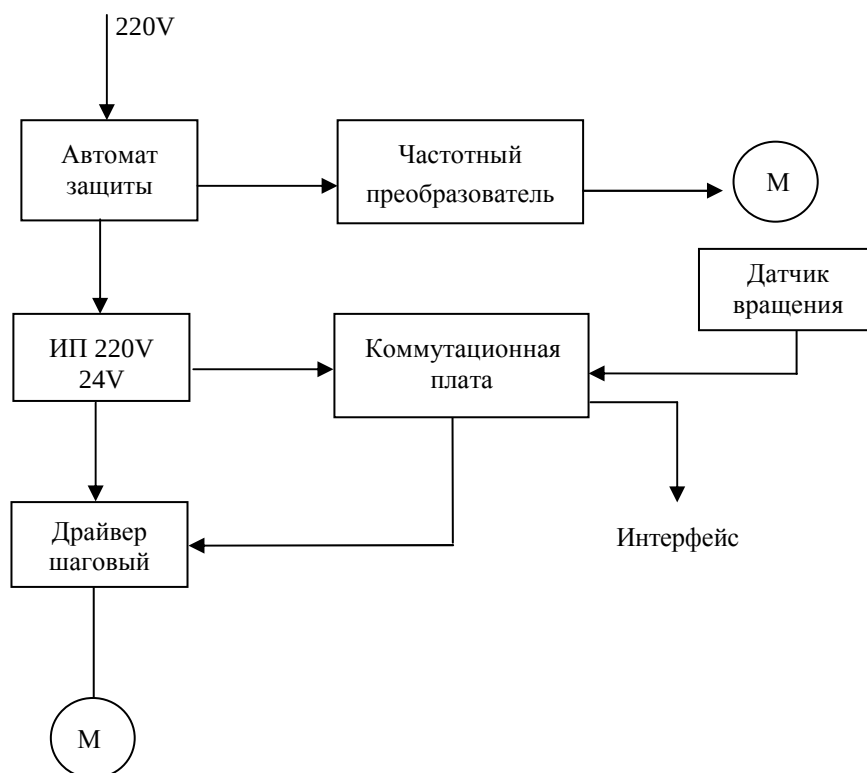


Рисунок 1. – Структурная схема стенда

Автоматы защиты служат для следующих целей:

- для отключения защиты частотного преобразователя и источника питания на 24 В. Через интерфейс от программируемого логического контроллера будут подаваться сигналы управления на коммутационную плату, на которую будет устанавливаться элементы гальванической развязки, например, оптопары и индикация режимов работы с помощью светодиодов;

- для передачи команд вращения частотного преобразователя и управления драйвером шагового двигателя. На коммутационную плату сигнал будет подаваться от датчика вращения, расположенного на асинхронном двигателе, и отправляться к интерфейсу ПЛК для подсчета количества вращений, например, шагового двигателя с целью измерения скорости и частоты его вращения.

Составные части стенда:

- частотный преобразователь Omron MX2;
- двигатель асинхронный;
- драйвер Autonics;
- шаговый двигатель.

Процесс реализации

Разработанная схема коммутационной платы будет содержать все необходимые органы управления по изменению управления стартом, стопом и основными характеристиками работы данной системы, а также обеспечивать гальваническую развязку интерфейса, подключаемого программируемого логического контроллера. Плата будет изготовлена на фольгированном стеклотекстолите, изготовлена с разработкой соответствующей электрической принципиальной схемой и сборочного чертежа. Также будет изготовлен макетный образец.

Заключение

В результате проведенного исследования разработана структурная схема стенда (состав), определен процесс реализации.

Работа над стендом продолжается. Осуществляется разработка документации для изготовления печатной платы.

ЛИТЕРАТУРА

1. AN1231 Plastic Ball Grid Array (PBGA) [Электронный ресурс]. – 2006. – Режим доступа: <http://www.analysisstech.com/downloads/PBGAExample.pdf>. – Дата доступа: 24.09.2017.

МАШИНОСТРОЕНИЕ

УДК621.7-52

ПРОГРАММИРОВАНИЕ ТЕХНОЛОГИЙ ВРЕЗАНИЯ И ОБРАБОТКИ КОНСТРУКТИВНЫХ ЭЛЕМЕНТОВ В ВИДЕ «КАРМАНОВ» ДЕТАЛЕЙ НА ФРЕЗЕРНЫХ СТАНКАХ С ЧПУ ПРИ ПОМОЩИ САМ-СИСТЕМ

Е.О. ЖИХОРЕВ, И.Н. ДОМКИН*(Представлено: д-р техн. наук, проф. Н.Н. ПОПОК; Р.С. ХМЕЛЬНИЦКИЙ)*

Рассмотрены возможности использования САМ-систем с целью создания управляющих программ для фрезерных станков с числовым программным управлением. Показаны преимущества использования данных систем в сфере металлообработки, описан принцип их работы. Приведены примеры использующихся САМ-систем, типов обрабатываемых с их помощью поверхностей на фрезерных станках с ЧПУ. На примере использования SolidCAM для создания управляющих программ при обработке «кармана», рассмотрены типы используемых циклов и технологии обработки.

Технические требования к точности и качеству деталей постоянно возрастают, что необходимо для различных отраслей производства: сельскохозяйственной, оборонной, промышленной. Наиболее ярко эти тенденции выражаются в промышленной сфере, а именно в машиностроении. В основном, именно благодаря машиностроению и продукции, которую она производит, осуществляется стабильная работа остальных сфер производства.

Ключевыми критериями любого производства является точность и качество обработанных поверхностей. В машиностроении точность определяется допусками на размеры, формы и взаимное расположение поверхностей и конструктивных элементов деталей [1]. Качество поверхностей деталей характеризуется их шероховатостью и физико-механическими свойствами.

Несоблюдение данных технических требований в процессе производства может в последствии привести к поломке или даже к ситуации, в которой могут пострадать люди.

Требования к точности часто достигают такого уровня, что без использования станков с числовым программным управлением (ЧПУ) их невозможно достичь.

Для обработки поверхностей деталей на станках с ЧПУ требуются специальные управляющие программы. Раньше эти программы писались вручную инженерами-технологами, но в настоящее время это зачастую невозможно из-за сложной конфигурации обрабатываемой поверхности или большого их количества, или большого числа изделий, которые требуется изготовить. Для этого с развитием информационных технологий были разработаны специальные САМ-системы.

Принцип работы САМ-систем. САМ-системы (computer-aided manufacturing) – компьютерная поддержка изготовления) автоматизируют расчеты траекторий перемещения инструмента для обработки на станках с ЧПУ и обеспечивают выдачу управляющих программ с помощью компьютера [2].

Принцип данных систем заключается в том, что технолог-программист избавляет себя от трудоемких математических расчетов и получает возможность значительно повысить скорость написания управляющих программ (УП).

Суть написания УП для обработки поверхности детали заключается в следующем: в специальных САД программах, наподобие КОМПАС-3D, SOLIDWORKS и других создается 2D- или 3D-модель детали. Затем при помощи САМ-системы на основе модели разрабатывается последовательность обработки поверхности, которую требуется изготовить. На последнем шаге САМ-система самостоятельно генерирует всю УП и выдает ее уже в виде G-кода готовой к вводу в систему ЧПУ станка. Благодаря тому, что все вычисления производятся компьютером, значительно увеличивается точность изготовления размеров.

В настоящее время для работы используется множество САМ-систем: GibbsCAM; MasterCAM (CNC Software, inc.); CAMConcept (EMCO MaierGes.m.b.H.); SolidCAM.

Особенно широко данные системы используются при фрезерной обработке, ввиду того, что существуют некоторые виды поверхностей (рис. 1), обработка которых более удобна и менее затратна на станках с ЧПУ. К таким поверхностям относятся: карманы различных конфигураций (рис. 1, а, 1, в), канавки сложной формы, а также сложные пространственные поверхности (рис. 1, б). Такие поверхности требуют высокой степени точности геометрии, которая часто не может быть достигнута на фрезерных станках, не оснащенных системой ЧПУ, а если и может быть достигнута, то для этого требуются большие затраты времени, высокая квалификация рабочего и т.д.

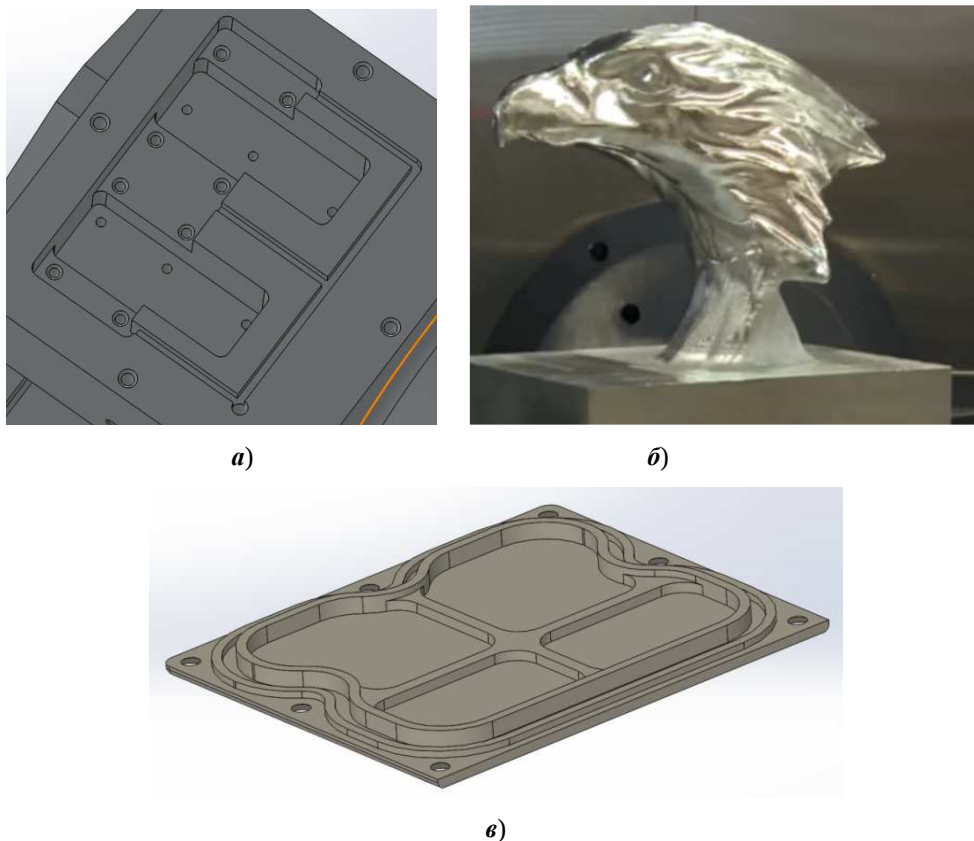


Рисунок 1. – Примеры поверхностей, обрабатываемых на фрезерных станках с ЧПУ

При определенной серийности производства станки с ЧПУ позволяют обрабатывать рассмотренные поверхности более качественно и с меньшими затратами. Достаточно написать программу обработки в САМ-системе и загрузить ее в систему станка, после чего произвести отладку станка, запустить ее и следить лишь за правильностью выполнения.

Рассмотрим **обработку карманов**, в частности **технологии и циклы врезания, программируемые САМ-системой**, а также **технологии и циклы, использующиеся при этом**.

Одной из возможностей обработки данных поверхностей является использование нескольких циклов врезания режущего инструмента в заготовку, а именно врезание с движением по одной (рис. 2, а), двум (под углом) (рис. 2, б) или трем координатам (спиральное или винтовое) (рис. 2, в) [3].

На рисунке 2 траектория врезания выделена утолщенной линией; штриховой – траектория снятия припуска; штриховой утолщенной – переходы между отдельными карманами.

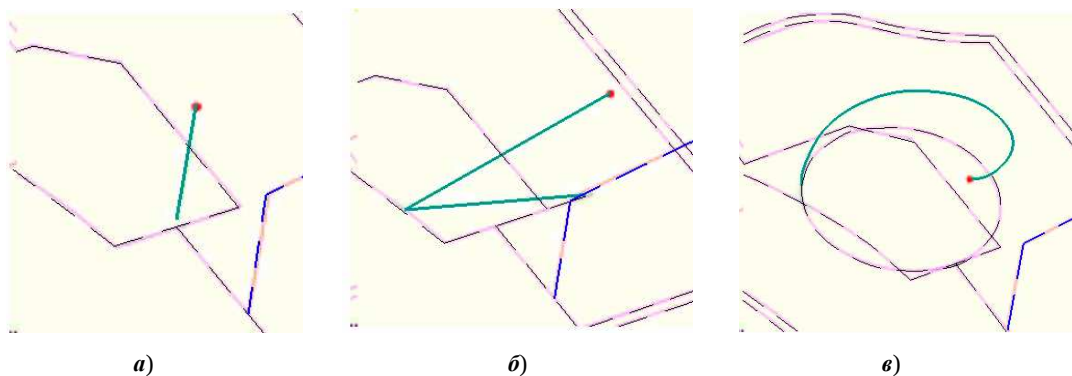


Рисунок 2. – Визуализация циклов врезания при фрезеровании кармана

Для первого и второго случая (рис. 2, а, 2, б) используется функция G01 – линейная интерполяция, для врезания по трем координатам (рис. 2, в) – G02, G03 – круговая/винтовая интерполяция по/против часовой стрелки.

Примеры программирования этапа врезания:

G02 X-54.Y-54.Z-10.I0. J4. Здесь координаты XYZ задают конечную точку, а I и J – координаты центра дуги, по которой движется инструмент.

G01 X50. Y-58. Здесь перемещение задано по прямой. Также возможно перемещение и по трем координатам, для чего требуется ввести координату Z.

Также существуют различные технологии и соответствующие им циклы для обработки основной части кармана: растровая (рис. 3, а), контурная (рис. 3, б), растровая + контурная (комбинированная) (рис. 3, в), а также плунжерное фрезерование (рис. 3, г) [3].

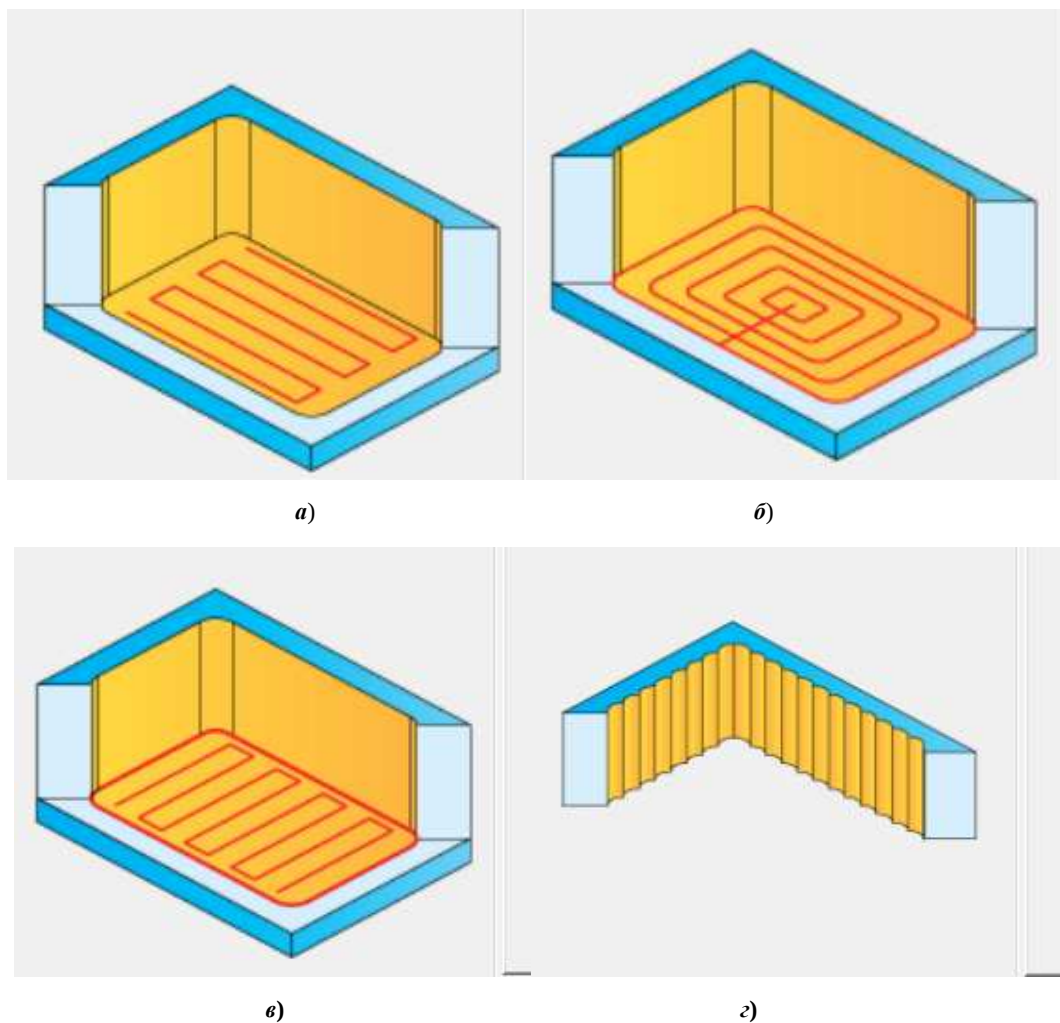


Рисунок 3. – Визуализация циклов обработки кармана

Отличие растровой и контурной технологии заключается в том, что в первом случае траектория движения инструмента задается точками, т.е. инструмент движется от точки к точке. При контурной обработке движение режущего инструмента определяется линиями. При выборе комбинированного способа обработки основная часть припуска снимается при помощи растровой технологии, а часть, оставшаяся около стенок кармана, – контурной.

Следует заметить, что так называемое плунжерное фрезерование в реальности является цекованием, так как по определению *фрезерование* – лезвийная обработка с вращательным главным движением резания при постоянном радиусе его траектории, сообщаемым инструменту, и хотя бы одним движением подачи, направленным перпендикулярно оси главного движения резания [4].

Как видно из приведенных примеров выше, САМ-системы позволяют комбинировать различные способы обработки. Кроме этого с их помощью возможно создание управляющих программ для таких видов обработки, как: фрезерование, центрование, сверление, зенкерование и развертывание отверстий, нарезание различных видов резьб. Также данные системы широко используются и в токарной обработке.

Заключение

Благодаря использованию САМ-систем инженер-технолог может выбирать необходимые ему способы и технологии обработки, зависящие от конкретной ситуации, серийности производства, сложности и конфигурации обрабатываемого кармана. Такое комбинирование различных циклов врезания и снятия основного припуска позволяет создавать новые управляющие программы.

Также использование САМ-систем с целью создания управляющих программ для станков с ЧПУ позволяет:

- 1) избавить технологов-программистов от необходимости производить многочисленные математические расчеты при разработке управляющих программ;
- 2) повысить точность и качество обработки;
- 3) обрабатывать детали и поверхности сложной конфигурации;
- 4) уменьшить влияние человеческого фактора в процессе производства;
- 5) закрепить за одним рабочим несколько станков, повысив таким образом показатель многостаночности;
- 6) уменьшить время обработки;
- 7) в определенных случаях уменьшить себестоимость изготовления детали;

ЛИТЕРАТУРА

1. Дунин-Барковский, И.В. Взаимозаменяемость, стандартизация и технические измерения : учебник / И.В. Дунин-Барковский. – М. : Изд-во стандартов, 1987. – 352 с.
2. Планета САМ [Электронный ресурс] // CAD/CAM / Что такое CAD и CAM. – Режим доступа: <http://planetacam.ru/college/learn/12-2/>. – Дата доступа: 02.10.2017.
3. Руководство SolidCAM 2006 R10.1. – 106 с.
4. База стандартов [Электронный ресурс] // Главная/ Общероссийский классификатор стандартов / Общие положения. Терминология. Стандартизация. Документация / Виды обработки резанием. Термины и определения общих понятий. – Режим доступа: <http://engenegr.ru/gost-25761-83>. – Дата доступа: 02.10.2017.

УДК621.7-52

АНАЛИЗ ОСОБЕННОСТЕЙ И ВОЗМОЖНОСТЕЙ РАЗЛИЧНЫХ СИСТЕМ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ В МАШИНОСТРОЕНИИ

И.Н. ДОМКИН, Е.О. ЖИХОВЕВ, В.С. АНИСИМОВ**(Представлено: д-р техн. наук, проф. Н.Н. ПОПОК; Р.С. ХМЕЛЬНИЦКИЙ)**

Рассматривается современное машиностроение, характеризующееся широким спектром проектных работ, сложностью конструкции, сжатыми сроками выполнения заказов, что вынуждает проводить работы на современном техническом уровне, с использованием интегрированных систем автоматизированного проектирования. Оптимизация ранее созданных технологических процессов необходима в силу долгосрочного их использования и стремления к уменьшению расходов на производство.

Компьютерное моделирование техпроцесса во многом ускоряет процесс изготовления деталей за счет выявления и устранения возможных ошибок на этапах конструирования и подготовки производства и изготовления изделия.

CAD/CAM являются компьютерной технологией для проектирования и выпуска рабочей документации. Программы CAD/CAM используются для проектирования изделий и программирования производственных процессов, в частности станков с числовым программным управлением (ЧПУ). В САМ используются модели и сборки, созданные в САД для формирования траекторий перемещения инструментов, которые управляют станками, создающими физические детали по проектам. Программное обеспечение CAD/CAM наиболее часто используется для обработки прототипов и готовых деталей [1].

САМ-системы (computer-aided manufacturing компьютерная поддержка изготовления) предназначены для проектирования обработки изделий на станках с ЧПУ (фрезерных, сверлильных, эрозионных, пробивных, токарных, шлифовальных и др.) и выдачи программ для этих станков. В настоящее время они являются практически единственным способом для изготовления сложнопрофильных деталей и сокращения цикла их производства. В САМ-системах используется трехмерная модель детали, созданная в САД-системе. Традиционно существует также деление CAD/CAM/CAE-систем на системы верхнего, среднего и нижнего уровня.

Примерами CAD/CAM-систем *верхнего уровня* являются Pro/Engineer, Unigraphics, CATIA (все они имеют расчетную часть CAE). Наиболее известными CAD/CAM-системами *среднего уровня* являются: ADEM; Cimatron (Cimatron Ltd.); Mastercam (CNC Software, Inc.); AutoCAD 2000; Powermill (DELCAM) и другие. К числу CAD/CAM-систем *среднего уровня* на основе ядра Parasolid принадлежат CADKEY 99 (CADKEY Corp.); SolidWorks (SolidWorks Corp.); Unigraphics Modeling (Unigraphics Solutions) и другие CAD-системы *нижнего уровня* (например, Medusa, TrueCAD, «Компас» и др.) применяются только при автоматизации чертежных работ.

По целевому назначению различают следующие системы автоматизированного проектирования (САПР) или подсистемы САПР, которые обеспечивают различные аспекты проектирования:

- **CAD** (англ. computer-aided design/drafting) – средства автоматизированного проектирования, в контексте указанной классификации термин обозначает средства систем автоматизированного проектирования, предназначенные для автоматизации двумерного и/или трехмерного геометрического проектирования, создания конструкторской и/или технологической документации, и САПР общего назначения;

- **CADD** (англ. computer-aided design and drafting) – средство проектирование и создание чертежей;

- **CAGD** (англ. computer-aided geometric design) – средство геометрического моделирования;

- **CAE** (англ. computer-aided engineering) – средства автоматизации инженерных расчетов, анализа и симуляции физических процессов, осуществляют динамическое моделирование, проверку и оптимизацию изделий;

- **CAA** (англ. computer-aided analysis) – подкласс средств CAE, используемых для компьютерного анализа;

- **CAPP** (англ. computer-aided process planning) – средства автоматизации планирования технологических процессов, применяемые на стыке систем САД и САМ.

Многие системы автоматизированного проектирования совмещают в себе решение задач, относящихся к различным аспектам проектирования CAD/CAM, CAD/CAE, CAD/CAE/CAM. Такие системы называют комплексными или интегрированными. С помощью САД-средств создается геометрическая модель изделия, которая используется в качестве входных данных в системах САМ и на основе которой в системах САЕ формируется требуемая для инженерного анализа модель исследуемого процесса.

В общем случае процесс работы в САМ-системе выглядит линейно. Начинается работа с подготовки геометрии; затем выбираются элементы, подлежащие обработке; после назначается стратегия; определяются параметры операции, режимы резания и инструмент; производится графическую проверку рассчитанных системой траекторий; запускается постпроцессирование и получается код управляющей программы (УП). За годы эволюции САМ этот процесс принципиально не изменился, лишь отдельные его фрагменты подверглись автоматизации или стали более функциональными и интерактивными. На любом из этапов процесса можно внести корректировки в операцию, например, изменить диаметр инструмента, глубину обработки, количество проходов и запустить пересчет операции с новыми параметрами. Такие итерации могут продолжаться многократно, пока программист и оператор на станке не будут удовлетворены результатом. Возрастающая сложность оборудования и необходимость сокращения времени подготовки УП сподвигла разработчиков САМ-систем на модернизацию устоявшегося процесса.

Новый подход заключается в интерактивном использовании станочных данных уже на начальном этапе проектирования операций обработки. Если сейчас программист взаимодействует с трехмерной моделью станка исключительно во время симуляции обработки, то в будущем он сможет производить виртуальную наладку, например, размещать приспособление на столе станка или позиционировать заготовку. Виртуальная наладка непосредственно влияет на принятие решений в процессе создания операций обработки и позволяет, к примеру, понять, хватит ли угла поворота рабочего стола с заготовкой или каким окажется качество поверхности обработанной детали с учетом данных об ускорении и замедлении рабочих органов станка. Причем речь идет не о режиме симуляции обработки, а о процессе, влияющим на создаваемые в САМ-системе траектории. При выборе стратегии обработки система будет способна предупредить, например, о невозможности использования данной стратегии, предложить максимально допустимые режимы резания или оценить выбранные, оптимизировать расчеты траектории с учетом кинематики станка (что особенно актуально для 5-ти осевого фрезерования) и, разумеется, помочь избежать коллизий. Более того, операции трансформации (копирование, поворот, смещение) над виртуальной деталью приведут к немедленному пересчету операций обработки.

Одним из наиболее перспективных аспектов применения САМ-систем является автоматизация подбора инструмента и режимов резания непосредственно из САМ-системы, которая передает параметры распознанных конструктивных элементов 3D-модели обрабатываемой детали в облако, а обратно получает список инструментов и рекомендованные значения режимов обработки. Причем нужно отметить, что импортированные инструменты служат не только для последующего расчета траекторий и верификации, но и снабжены каталожными номерами, что значительно облегчает работу технолога, отвечающего за выбор и приобретение инструмента, необходимого для выполнения текущего производственного задания.

Неотъемлемой частью САМ-систем является постпроцессирование. Постпроцессор – программа, которая преобразует файл траектории движения инструмента и технологических команд (промежуточный файл), сформированный САМ-системой, в файл УП в соответствии с требованиями конкретного комплекса «станок – система ЧПУ». В ряде отечественных систем постпроцессоры какое-то время назывались паспортами.

Для того чтобы абстрагироваться от большого разнообразия станков, систем ЧПУ и языков программирования обработки, САМ-система генерирует промежуточный файл, содержащий информацию о траектории, угле поворота инструмента (в случае многокоординатной обработки) и обобщенные команды управления станком. Обычно этот промежуточный файл называется CL-файлом (CutterLocation) или CLDATA-файлом. Далее в работу вступает постпроцессор. Он преобразует этот промежуточный файл в программу обработки в строгом соответствии с форматом программирования конкретного станка с ЧПУ.

Такая технология позволяет программисту во время проектирования обработки в САМ-системе не задумываться о том, на какой конкретно станок попадет УП и каков будет ее формат. Ему необходимо лишь выбрать постпроцессор, соответствующий определенному станку с ЧПУ, и тот возьмет на себя всю работу по созданию программы обработки определенного формата [2].

Постпроцессирование, пожалуй, одна из самых консервативных областей САМ. Тем не менее и здесь можно проследить определенную эволюцию.

К сожалению, разработчики САМ-систем не придерживаются единого стандарта для формирования промежуточных CL-файлов. В результате текстовый файл с описанием алгоритмов преобразования (постпроцессор) одной САМ-системы будет абсолютно бесполезен для использования внутри другой САМ-системы. Для разработки даже такого текстового постпроцессора «с нуля» необходимы глубокие знания самой системы и принципов преобразования исходных данных. Поэтому пользователю предоставляют набор базовых постпроцессоров с редактируемыми переменными. Сегодня наиболее эффективным и простым решением проблем постпроцессирования на предприятии является использование универсального постпроцессора, так называемого генератора постпроцессоров. В таблице 1 представлены основные характеристики некоторых наиболее распространенных САМ-систем.

Таблица 1. – Характеристики САМ-систем

Программа	Страна (город)	Компания	Самостоятельная/ в составе	Верификация	Тип промышленности
T-FLEX CAD	Россия (Москва)	ЗАО «ТОП Системы»	–/+	–	Общее машиностроение
TEBIS	Германия	TebisTechnische Information system AG	–/+	+	Общее машиностроение
VISI-Series	Мексика	Vero Software	–/+	–	Аэрокосмическая, автомобильная промышленность
EdgeCAM	Англия	Vero Software	+/-	+	Общее машиностроение
SprusCAM	Россия	«Спрут-Технология»	+/-	+	Общее машиностроение
UNIGRAPHIC SNX	США – Германия	United Computing-Siemens PLM Software	–/+	+	Судостроение, двигателестроение
ESPRIT	США	DP Technology	–/+	+	Общее машиностроение, аэрокосмическая промышленность
MasterCAM	США	CNC Software	–/+	+	Общее машиностроение, деревообработка
SolidCam	США	CNC Software	–/+	+	Общее машиностроение
Cimatron	Израиль	Cimatron Ltd.	–/+	+	Автомобильная, авиационная, космическая, медицинская промышленность, производство электроники
CATIA	Франция	DassaultSystemes	–/+	+	Аэрокосмическая, автомобильная, радиоэлектронная, судостроительная
VX	Китай	VXCorp.- ZW Soft	–/+	+	Общее машиностроение
Emco CamConcept	Германия	EMCO	+/-	–	Общее машиностроение
Powersolutions	Великобритания	Small Heath	–/+	+	Общее машиностроение
ADEM	Россия (Москва)	АДЕМ	–/+	+	Авиационная, аэрокосмическая, машиностроительная промышленность

Как видно из таблицы 1, наиболее известными программами на данный момент являются 3 российских программы и 11 иностранных (США – 3, Великобритания – 2, Германия – 2, Мексика, Израиль, Китай, Франция). Среди них самостоятельных (автономных) – 3 наименования, остальные 11 находятся в составе PLM (систем управления жизненным циклом изделия).

В таблице 2 представлены возможности САМ-систем, которые свидетельствуют о применении 5-осевой обработке на токарных, сверлильных, фрезерных и электроэрозионных операциях.

Таблица 2. – Основные возможности САМ систем

Программа	Возможности	Примечания
T-FLEX CAD	Электроэрозионная обработка 2–5-координатное фрезерование	
TEBIS	Обработка 3 + 2 оси, точение, фрезерование	Сканирование поверхностей для анализа геометрии, интерактивное управление инструментами
VISI-Series	2–5-координатная механическая обработка, электроэрозионная обработка	Удобное проектирование форм для литья

Окончание таблицы 2

Программа	Возможности	Примечания
EdgeCAM	2,5–5-координатная фрезерная обработка, токарная обработка, электроэрозионная обработка	Мастер постпроцессоров, библиотека инструментов, исключение холостых проходов инструмента
SprusCAM	2,5-5-координатная фрезерная обработка, токарная обработка токарно-фрезерная обработка, электроэрозионная обработка, плазменная, лазерная, газовая	Возможна разработка программ для управления промышленными роботами
UNIGRAPHIC SNX	Токарная обработка, фрезерная обработка на 3-5-осевых станках, токарно-фрезерная обработка	Поддержка высокоскоростного фрезерования
ESPRIT	Токарная обработка на многоосевых станках, 2,5-осевое фрезерование, циркулярное фрезерование по 4-й оси	Реалистичная точная визуализация
MasterCAM	2,5-5-координатная фрезерная обработка, токарная обработка токарно-фрезерная обработка, электроэрозионная обработка	Возможно проектирование и создание объемных художественных объектов из плоских шаблонов. Модуль для деревообработки. Генерирование ЧПУ программы
SOLIDCAM	2,5-5-координатная фрезерная обработка, токарная обработка токарно-фрезерная обработка, электроэрозионная обработка Сверлильная обработка	Возможно проектирование и создание объемных художественных объектов из плоских шаблонов. Реалистичная точная визуализация. Генерирование ЧПУ программы
Cimatron	Токарная, токарно-фрезерная, проволочная эрозионная обработка 2,4,5-координатная фрезерная обработка	Поддержка станков с противощпинделем, генератор постпроцессоров
CATIA	Токарная обработка, сверлильная обработка	Большое предназначение моделированию деталей и кинематики
VX	5-осевая обработка, гравировка по 3-м осям	
Emco CamConcept	Токарная, фрезерная, сверлильная	Генерирование ЧПУ программы
Powersolutions	Токарная, фрезерная, сверлильная обработка	
ADEM	Фрезерование с произвольным числом управляемых осей, токарная, токарно-фрезерная обработка, гравировка, обработка на прессах с ЧПУ	

Таким образом, CAD/CAM-технологии позволяют добиться двух важных результатов – сократить время изготовления продукции и значительно снизить влияние человеческого фактора (а значит, ошибок). Самостоятельное, не связанное между собой функционирование систем CAD и CAM дает эффект, размер которого может быть существенно увеличен интеграцией этих систем посредством CAPP. Такая интегрированная система CAD/CAM на информационном уровне поддерживается единой базой данных, в которой хранится трехмерная математическая модель изделия, т.е. информация о его структуре и геометрии (как результат проектирования в системе CAD), о технологии изготовления (как результат системы CAPP) и УП для оборудования с ЧПУ (как исходная информация для обработки в системе CAM на оборудовании с ЧПУ) [3–5].

ЛИТЕРАТУРА

1. CAD/CAM-системы [Электронный ресурс]. – Режим доступа: <https://www.autodesk.ru/solutions/cad-cam>. – Дата доступа: 16.09.2015.
2. Будущее CAM-систем [Электронный ресурс]. – Режим доступа: http://isicad.ru/ru/articles.php?article_num=18313/. – Дата доступа: 16.09.2015.
3. Смелов, В.Г. Пути 3D-параметризации лопаток ГТД для использования с CAD/CAM/CAE пакетах / В.Г. Смелов, И.Л. Шитарев // Вестн. Самар. гос. аэрокосм. ун-та. – 2008. – № 3. – С. 44–47.
4. Прогрессивные технологии моделирования, оптимизации и интеллектуальной автоматизации этапов жизненного цикла авиационных двигателей: / А.В. Богуслаев [и др.] ; Запорож. нац. техн. ун-т. – Запорожье : Мотор-Сич, 2009. – 468 с.
5. Невлюдов, И.Ш. Использование CAD/CAM/CAE/CAPP при формировании управляющих программ для станков с ЧПУ / И.Ш. Невлюдов, С.С. Великодный, М.А. Омаров // Вост.-Европ. журнал передовых технологий. – 2010. – № 2/2 (44). – С. 37–44.

УДК 621.924.1

**К ВОПРОСУ О МОДЕРНИЗАЦИИ СТАНОЧНОГО ОБОРУДОВАНИЯ
ДЛЯ ВЫСОКОСКОРОСТНОЙ ОБРАБОТКИ РЕЗАНИЕМ****К.В. КОРБА, В.С. АНИСИМОВ****(Представлено: д-р техн. наук, проф. Н.Н. ПОПОК; Р.С. ХМЕЛЬНИЦКИЙ)**

Рассмотрены возможные варианты модернизации станочного оборудования для высокоскоростной обработки. Предложено расширить технологические возможности универсального заточного станка модели ЗМ642, в части увеличения скорости вращения шпинделя с 6300 мин^{-1} до 12000 мин^{-1} . В электрические цепи приводов инструмента и детали включены датчики обратной связи по скорости, программируемый логический контроллер, коммутационные устройства, датчики скорости и панель оператора.

В общем случае можно выделить три основных варианта модернизации оборудования [1]:

1) *капитальный ремонт станка с доведением норм точности до паспортных значений*, с полной заменой систем управления и контроля оборудования. Возможности проведения подобных работ в значительной степени ограничены дефицитом, в первую очередь, квалифицированных специалистов, выполняющих ручные операции, например шабрение, а также опытных слесарей-сборщиков, инженеров-электронщиков, способных разобраться в сложном оборудовании, тем более что на него иногда отсутствуют документация и чертежи;

2) *модернизация оборудования с заменой части его функциональных узлов, заменой систем управления и измерения с расширением его технологических возможностей*. Это наиболее сложный вид ремонта. При этом от оборудования, по сути, используется станина и крупные корпусные узлы. При такой модернизации производится полная разборка станка и очистка всех деталей. Недостатком данного метода, как и предыдущего, является высокие требования к техническому персоналу, задействованному при такой модернизации, а также к производству, в условиях которого производится модернизация;

3) *замена систем управления и измерительных систем на системы, построенные на современной технической базе. Наиболее доступный в технологическом плане способ модернизации*. Данный вид модернизации не требует каких-либо значительных работ, связанных с восстановлением механической части оборудования, и в таком случае отпадают требования к оснащению производства, на котором производится модернизация. При данном типе модернизации производится полная или частичная замена системы управления, измерительной системы станка. Производится замена электропроводки и при необходимости двигателей.

В настоящее время в характеристиках систем числового программного управления (СЧПУ) часто выделяют две характеристики [2; 3]:

- возможность использовать его для управления высокоскоростной обработкой (ВСО) и соответствующим оборудованием;
- принятая схема подготовки управляющих программ.

Для достижения требуемой точности обработки, особенно финишной, требуются частые проходы инструмента с небольшим шагом. Следовательно, программируемые траектории инструмента являются сложными и многоточечными, представленными в управляющих программах большим количеством кадров.

Для обеспечения непрерывного движения инструмента в устройствах числового программного управления (УЧПУ) нужны высокие скорости обработки данных, как минимум на 150...200 блоков вперед, чтобы вычислять изменения величины подачи при подходе инструмента к острым углам (или другим подобным препятствиям) и отходе от них.

Для повышения качества поверхности и снижения нагрузок на инструмент необходимо, чтобы закон изменения величины подачи имел плавный, «колоколообразный» вид, поскольку причиной снижения качественных характеристик процесса являются слишком резкие ускорения при движениях по траекториям с углами.

Большое количество моделей УЧПУ, выпускаемых разными фирмами, требует внимательного изучения при выборе конкретного варианта для использования на практике. Естественно, при прочих равных характеристиках важнейшими факторами являются цена и гарантии стабильности работы, последние могут быть определяющими.

В представляемой работе поставлена задача создания на базе универсально-заточного станка модели ЗМ642 экспериментальной установки для исследования скорости обработки сферических поверхностей деталей [4]. Для получения скорости шпинделя не менее 12000 мин^{-1} необходимо увеличить

частоту напряжения питания с 50 Гц до частоты, найденной из соотношения $F_{\text{пит}} = 12000 / (6300 / 50)$, где 12000 мин^{-1} – необходимая скорость вращения шпинделя; 6300 мин^{-1} – номинальная скорость шпинделя при питании двигателя привода шпинделя напряжением промышленной частоты; 50 – частота напряжения питающей сети, при этом $F_{\text{пит}} = 95,23 \text{ Гц}$.

После изучения технических характеристик частотных преобразователей, представленных на рынке [5], выбраны частотные преобразователи фирмы Yaskawa серии V1000, так как у них имеется возможность ввода на аналоговый выход сигнала пропорционального потребляемого асинхронным двигателем. Были выбраны модели VZA21P58AA для управления двигателем шпинделя, и VZA20P48AA для управления двигателем приспособления наружного круглого шлифования 3Е642Е.П17.

Выбран программируемый логический контроллер (ПЛК) Siemens SIMATIC S7-200 CPU24NP, который удовлетворяет предъявляемые требования: напряжение питания 24 В; 14 дискретных входов на 24 В; 10 дискретных выходов на 24 В/0,75 А; 2 аналоговых входа; 1 аналоговый выход; память программ 16 Кбайт; память данных 10 Кбайт; 2 PPI/ MPI/ свободно программируемых порта; подключение до 7 модулей расширения.

В качестве устройства человеко-машинного интерфейса используется панель оператора OP27 фирмы Siemens, представленная на рисунке 1.

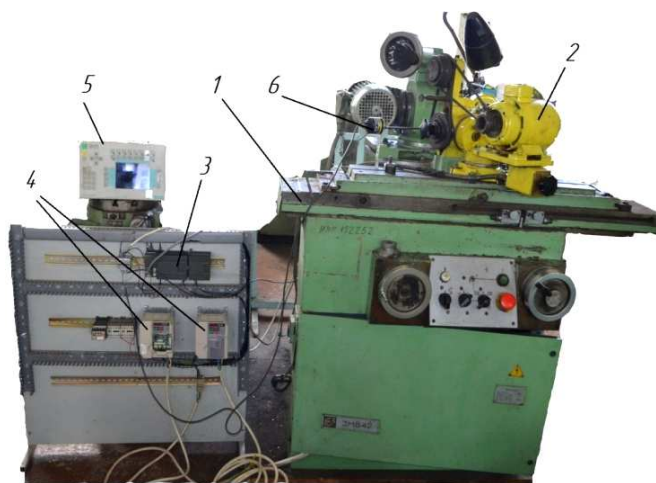


Рисунок 1. – Панель оператора OP27 фирмы Siemens

На рисунке 2 показан универсально-заточной станок модели 3М642 с экспериментальным стендом.

Элементы схемы испытательного стенда смонтированы на передней панели несущей конструкции, предназначенной для сборки экспериментальных устройств.

На панели расположены: в верхнем ряду слева направо клеммная колодка XT4 для подключения датчиков обратной связи по скорости программируемый логический контроллер Siemens S7-200 CPU224NP с модулем расширения EM235.



1 – станок модели 3М642; 2 – приспособление для наружного круглого шлифования 3Е642Е.П17; 3 – промышленный контроллер; 4 – частотные преобразователи; 5 – операторская панель; 6 – энкодер

Рисунок 2. – Универсально-заточной станок модели 3М642 с экспериментальным стендом

В нижнем ряду расположены коммутирующие устройства QF2-QF4 и частотные преобразователи F11и F12, которые предназначены для управления двигателями привода шпинделя и привода детали соответственно. Здесь же расположен блок питания 24 VDC для питания программируемого логического контроллера Siemens S7-200 CPU224NP с модулем расширения EM235, датчики скорости и панели оператора Siemens OP27.

Подытоживая проведенное исследование можно сделать следующие **выводы**:

1. Рассмотрены возможные варианты модернизации станочного оборудования для повышения скорости обработки резанием, измерения мощности приводов и управления частотой вращения. Установлено, что наиболее технологичным способом модернизации является замена систем управления и измерительных систем на системы, построенные на современной технической базе.

2. Определена частота напряжения питающей сети равная 95,23 Гц, которая обеспечивает увеличение частоты вращения шпинделя станка с 6300 мин⁻¹ до 12000 мин⁻¹. На основе анализа характеристик выбраны частотные преобразователи фирмы Yaskawa серии V1000, программируемый логический контроллер фирмы Siemens SIMATIC S7-200 CPU24NP, панель оператора OP27 фирмы Siemens.

3. Произведена модернизация универсального заточного станка модели 3М642 путем включения в электрические цепи приводов инструмента и детали датчиков обратной связи по скорости, программируемого логического контроллера, коммутационных устройств, для управления двигателями приводов, блока питания, датчика скорости и панели оператора. Это позволяет без сложных систем ЧПУ управлять бесступенчато частотой вращения приводов и измерять мощность резания.

ЛИТЕРАТУРА

1. Потапов, В.А. Опыт американских и германских фирм по ремонту, восстановлению и модернизации станочного оборудования [Электронный ресурс] / В.А. Потапов. – Режим доступа: http://stanki-katalog.ru/st_4.htm.
2. Элементы управления серии «Логика-И» / В.Л. Рейзин [и др.]. – М. : Энергоатомиздат, 1984. – 67 с.
3. Серебrenицкий, П.П. Некоторые особенности высокоскоростной механической обработки / П.П. Серебrenицкий // Металлообработка. – 2007. – № 4. – С. 6–15.
4. Универсально-заточной станок мод. 3М642, 3М642Е, 3М642Е-1 : Руководство по эксплуатации 3М642Е.00.000 РЭ Станкоимпорт, 1982. – 51 с.
5. General-Purpose Incremental Rotary Encoder With Large Shaft 40 dia. Omron [Электронный ресурс]. – Режим доступа: <http://www.sas.by/upload/iblock/205/205f69ae0db76d715e8127413c42a253.pdf>.

УДК 621.785

СПОСОБЫ СНИЖЕНИЯ НАПРЯЖЕНИЯ СВАРНОГО СОЕДИНЕНИЯ

С.В. СКУРЬЯТ, К.О. РЫКТЕР
(Представлено: М.В. СЕМЕНЧЕНКО)

Показана актуальность снижения напряжения сварных соединений путем термической обработки. Представлен обзор возможных способов термического воздействия на различных этапах изготовления сварного изделия, используемых источников нагрева и режимов обработки. Предложен способ отжига сварных изделий в условия электроконтактного нагрева в режиме термоциклирования.

Повышение качества сварных изделий путем снижения образовавшихся напряжений с помощью термической обработки является одним из актуальных направлений. Для достижения оптимального результата требуется правильный подбор способа воздействия, оборудования и режимов. В зависимости от особенностей изделия и состава используемого материала предпочтение может быть отдано нагреву:

- деталей перед сваркой, способствующей снижению скорости охлаждения сварного шва;
- в процессе сварки (сопутствующий нагрев);
- готового сварного соединения.

К основным технологическим факторам термической обработки, позволяющим снизить напряжения сварного соединения, следует отнести:

- температуру нагрева;
- время и скорость нагрева;
- продолжительность выдержки при заданной температуре;
- продолжительность «паузы» между формированием сварного шва и термическим воздействием;
- скорость охлаждения.

Для снятия внутренних напряжений изделия, изготовленные из низкоуглеродистых и среднеуглеродистых сталей, подвергают низкому отжигу при температуре 600...680 °С в печи [1]. Продолжительность выдержки зависит толщины металла и выбирается из расчета 2,5 мин на 1 мм. Охлаждение производится вместе с печью или на воздухе.

В некоторых случаях прибегают к полному отжигу, позволяющему не только устранить внутренние напряжения, но и улучшить структуру материала [2]. Металл становится более пластичным. Обработку осуществляют при температуре 820...930 °С, выдерживая не менее 30 минут. Охлаждение с печью выполняют до температуры 300 °С, а затем на воздухе. Преимуществом описанного способа термического воздействия является формирование однородной структуры сварного шва или околошовной зоны. Однако такое изделие, как правило, подвергается улучшению для измельчения зерна и улучшения эксплуатационных характеристик.

Нормализацию сварных изделий выполняют при 850...900 °С. Последующее охлаждение на воздухе способствует измельчению зерна металла шва и околошовной зоны. Материал становится тверже и прочнее. Если углеродистый эквивалент больше 0,38, велика вероятность появления в сварном соединении холодных трещин из-за образования закалочных структур и возникновения растягивающих напряжений.

Кроме печного нагрева для снижения напряжения сварного шва может использоваться [3]:

- пламя газовой горелки. Способ предполагает использование тепла, образующегося в результате сгорания газовой смеси. Учитывая, что тепло подводится извне изделия, температурный градиент по толщине материала может достигать больших значений. В среднем колебания по толщине изделия составляет 1 °С/мм. Также из-за соприкосновения пламени с металлической поверхностью протекают окислительные процессы;

- электрическая дуга, сформированная при использовании сварки неплавящимся электродом (вольфрамовым или графитовым);

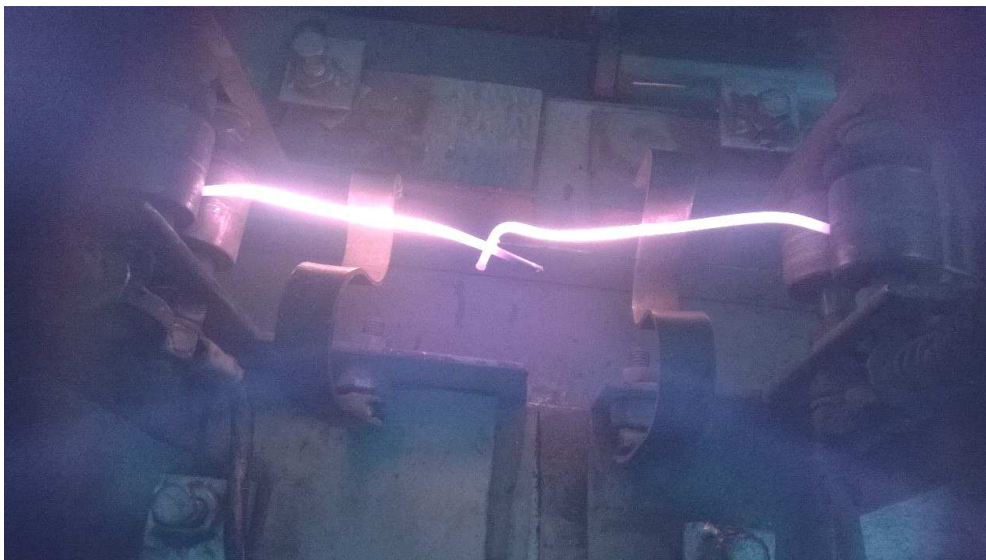
- накладной ТЭН. Способ подходит для нагрева изделий различного размера, однако отличается неравномерностью нагрева по толщине и периметру сварного соединения. Величина температурного перепада составляет 1 °С/мм;

- токи высокой частоты (ТВЧ). Характеризуется незначительным температурным градиентом, однако предполагает использование специального громоздкого электрооборудования.

Для условий единичного производства представляется перспективным использование электроконтактного нагрева, позволяющего за несколько секунд разогреть сварное изделие до требуемой температуры. Обработка может осуществляться в режиме непрерывного термического воздействия или термоциклирования. Последнее предполагает непрерывный нагрев и охлаждение по заранее выбранным режи-

мам, определяемым геометрическими параметрами сварного соединения и химическим составом материала изделия.

Нами была проведена серия опытов, направленных на снижение напряжения сварного шва в условиях электроконтактного нагрева (рисунок). Отжиг выполнялся на лабораторной установке, позволяющей реализовать режим термоциклирования [4].



Электроконтактный нагрев сварного соединения в режиме термоциклирования

Заключение

Полученные результаты свидетельствуют о том, что способ актуален для этапа предварительного и сопутствующего подогрева сварного шва, а термическая обработка в условия электроконтактного нагрева является перспективным направлением снижения напряжений сварных соединений.

ЛИТЕРАТУРА

1. Режимы термообработки стали [Электронный ресурс] – 2011. – Режим доступа: <http://nt-welding.ru/articles/rezhimy-termoobrabotki-stali/>. – Дата доступа: 26.09.2017.
2. Термическая обработка сварных соединений [Электронный ресурс] – 2014. – Режим доступа: <http://www.xiron.ru/content/view/30358/28/>. – Дата доступа: 26.09.2017.
3. Термическая обработка сварных швов и соединений [Электронный ресурс]. – 2014. – Режим доступа: <http://elisit.ru/materiali/termoobrabotka-svarnyx-shvov/>. – Дата доступа: 27.09.2017.
4. Установка для электротермической обработки проволоки: пат. на полезную модель № 696 МПК 7 C21D 1/40 / В.М. Константинов, А.С. Губанов, С.Н. Абраменко, М.В. Семенченко ; заявитель Полоц. гос. ун-т. – № и 20020065 ; заявл. 05.03.2002. ; опубл. 30.12.2002.

УДК 621.891:539.217.1

ВАКАНСИОННЫЙ МЕХАНИЗМ ЗАЛЕЧИВАНИЯ ТРЕЩИН В ТВЕРДОМ ТЕЛЕ

А.С. ВАБИЩЕВИЧ

(Представлено: канд. техн. наук, доц. В.Э. ЗАВИСТОВСКИЙ)

В рамках структурно-диффузионного механизма взаимодействия дефектов с трещинами в твердом теле рассмотрена задача о вакансионном процессе залечивания трещин в металле с учетом зависимости коэффициента диффузии дефектов от температуры. Показано, что энергия активации диффузии пор в металле существенным образом влияет на скорость залечивания трещин.

Дефекты в твердых телах, в частности в металлах, могут быть различного типа и происхождения. Дефектная структура твердых тел формируется уже на этапе получения самого материала. Известно [1; 2], что в процессе изготовления из него изделий и при эксплуатации готовых изделий она может существенно трансформироваться. К наиболее типичным дефектам кристаллической структуры относятся: точечные дефекты (вакансии, междоузельные атомы); линейные дефекты (дислокации и их петли); поверхностные дефекты (границы зерен, дефекты упаковки); объемные дефекты (поры и включения).

В процессе технологической и механической обработки материала возникают характерные дефекты: трещины, поры, включения. Процесс разрушения начинается с образования трещин, их увеличением и заканчивается макроскопическим разрушением материала или конструкции на отдельные части. В материаловедении металлов состояние поверхности и зарождение трещин во многом определяет сопротивление материалов деформации и разрушению. При определенных условиях трещины в металле способны залечиваться (уменьшать свой объем), что приводит к улучшению эксплуатационных характеристик материала.

В данной работе рассматривается возможный результат залечивания трещин в рамках модели структурно-диффузионного механизма, предложенной ранее авторами [3]. Согласно указанной модели, трещины являются стоками для точечных дефектов и микропор в твердом теле. Процесс залечивания (разрастания) связан с потоком микропор (вакансий) от трещины (к трещине). Ранее установлено [3], что концентрация вакансий у поверхности малых трещин (пор) выше, чем у больших, и, следовательно, появляется поток вакансий от малых трещин к большим. Происходит залечивание малых трещин.

Залечивание трещин может также сопровождаться диффузией атомов матрицы либо примесей по вакансиям (микропорам) к трещинам. В настоящей работе в качестве диффундирующего элемента выбран хром, который используют для увеличения прочности и уменьшения коррозии железа.

Модель вакансионного механизма взаимодействия пор с трещиной. Поток микропор на поверхность единицы длины трещины f рассчитывается по формуле [3]:

$$f = \frac{\gamma n_1}{R} \sqrt{\frac{D}{n_2}} \frac{K_1(z)}{K_0(z)}, \quad (1)$$

где

$$\gamma = \frac{2\alpha\Omega}{kT}, \quad z = R\sqrt{\frac{n_2}{D}},$$

R – радиус трещины; D – эффективный коэффициент диффузии для дефектов данного типа; n_1, n_2 – константы; α – удельная поверхностная энергия; Ω – характерный объем одной микропоры (дефекта); T – абсолютная температура; $K_i(z)$ – цилиндрические функции Макдональда i -го порядка.

Учтем, что температурная зависимость коэффициента диффузии дефектов может быть описана известным соотношением:

$$D = D_0 e^{-\frac{E_a}{kT}}, \quad (2)$$

где E_a – энергия активации диффузии дефекта; D_0 – постоянная величина; k – постоянная Больцмана.

Таким образом, поток микропор оказывается сложной функцией многих параметров: температуры, коэффициента диффузии, поверхностной энергии, радиуса трещины.

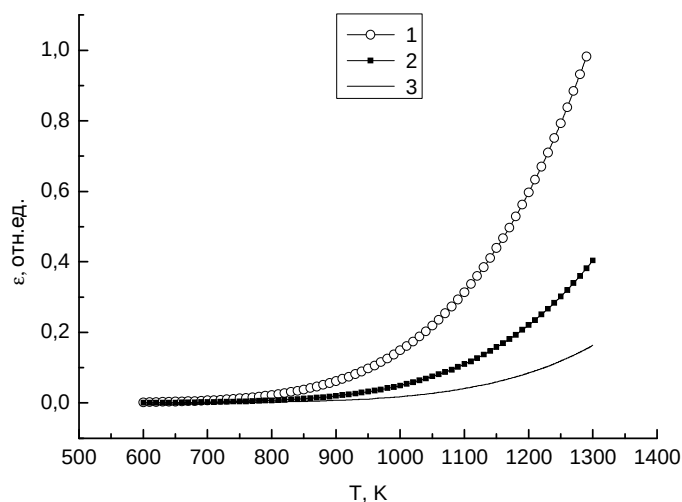
В настоящей модели анализируется диффузия микропор в объеме, прилегающем к открытой трещине цилиндрической формы. Изменение объема трещины ΔV длиной h за время t определяется по формуле:

$$\Delta V = 2\pi R h f t \Omega. \quad (3)$$

Оценка процесса залечивания трещины в настоящей модели проводится на основании расчета относительного изменения объема трещины (относительной скорости залечивания трещины):

$$\varepsilon = \Delta V / V. \quad (4)$$

На рисунке 1 представлены зависимости относительного изменения объема трещины от температуры для различных значений энергии активации диффузии микропор. Известно [2; 4], что в металлах энергия активации для вакансий лежит в интервале 0,9...2,0 эВ.



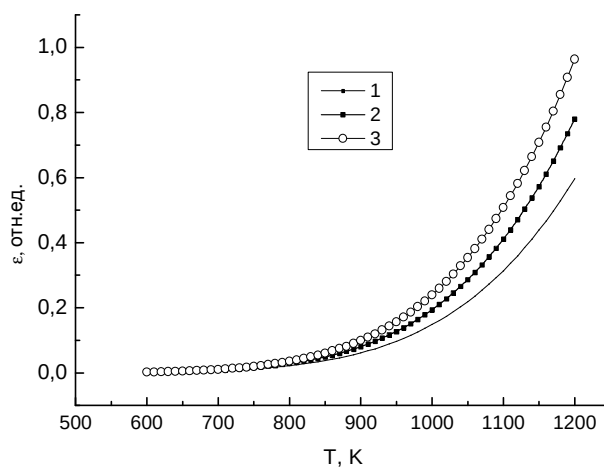
1 – 1, 2 – 1,13; 3 – 1,25
 $D_0 = 5 \cdot 10^{-6} \text{ м}^2/\text{с}$, $t = 20 \text{ ч}$

Рисунок 1. – Зависимость относительного изменения объема трещины от температуры при различных энергиях активации пор, ЭВ

Как видно из рисунка 1 изменение энергии активации пор в небольшом интервале 1...1,25 эВ существенным образом влияет на параметр залечивания. Кроме этого, скорость процесса залечивания зависит от температуры. С ростом температуры резко возрастает отличие в скорости залечивания в исследуемом интервале энергий.

Предполагается, что поток примесных атомов в процессе залечивания должен быть на порядок меньше вакансионного механизма, поскольку энергия миграции для примесей E_a лежит в широких пределах от 1 до 5 эВ. Таким образом, основное влияние пор на залечивание трещин связано с низким значением энергии активации диффузии пор.

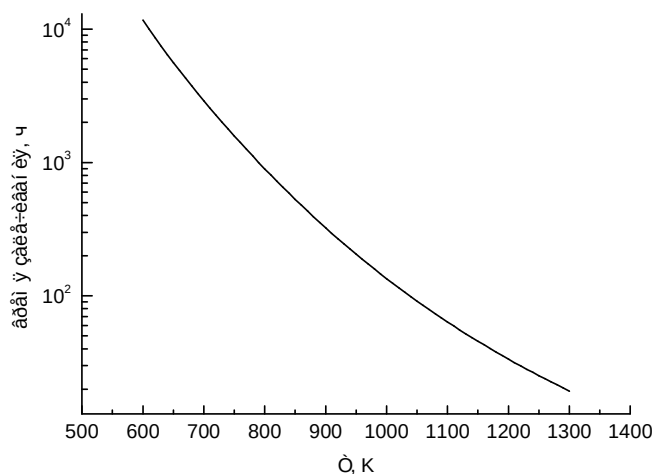
Кроме энергии активации на величину потока пор (скорость залечивания трещин) оказывает значение поверхностной энергии для металла (рис. 2).



1 – 1,3; 2 – 1,7; 3 – 2,1
 $D_0 = 5 \cdot 10^{-6} \text{ м}^2/\text{с}$, $t = 20 \text{ ч}$

Рисунок 2. – Зависимость относительного изменения объема трещины от температуры при различных значениях поверхностной энергии, Дж/м²

На рисунке 3 приведен график зависимости полного времени залечивания трещин от температуры. Видно, наименьшее время залечивания характерно для высоких температур 900...1000 К.



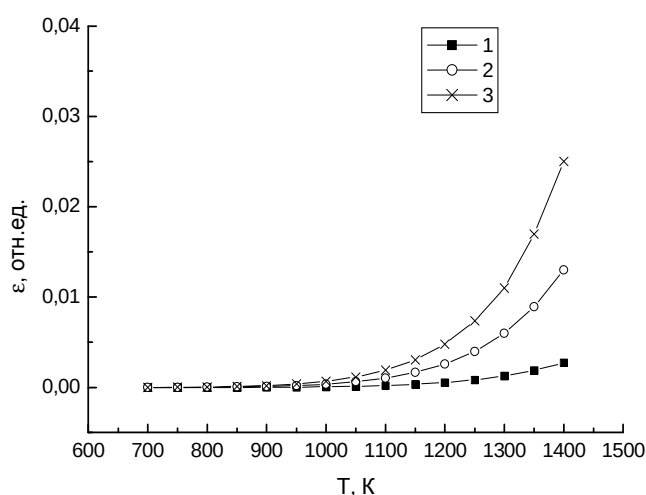
$$E_a = 1 \text{ эВ}, D_0 = 5 \cdot 10^{-6} \text{ м}^2/\text{с}$$

Рисунок 3. – Зависимость времени полного залечивания трещины от температуры

Моделирование диффузии хрома в железе. Хром в железе образует твердый раствор замещения. Залечивание трещин может сопровождаться диффузией атомов хрома по вакансиям (микропорам) к трещинам. Рассмотрим диффузию хрома в α -Fe в рамках вакансионного механизма. Поток атомов на поверхность единицы длины трещины f можно оценить по формуле (1), где D – эффективный коэффициент диффузии для атомов хрома. Учтем температурную зависимость коэффициента диффузии хрома (формула (2)), где E_a – энергия активации диффузии атомов хрома.

В качестве модели рассмотрим диффузию атомов Cr в объеме, прилегающем к открытой цилиндрической трещине. Изменение объема трещины и относительное изменение объема трещины определим по формулам (3) и (4).

На рисунке 4 представлены результаты моделирования зависимости относительного изменения объема трещины при диффузии атомов хрома.



$$1 - 2,4; 2 - 2,2; 3 - 2,0; D_0 = 0,85 \cdot 10^{-4} \text{ м}^2/\text{с}$$

Время отжига $t = 20 \text{ ч}$

Рисунок 4. – Зависимость относительного изменения объема трещины от температуры отжига при различных энергиях активации диффузии хрома, эВ

Для хрома в α -Fe энергия активации лежит в пределах от 2,17 до 2,4 эВ, а частотный фактор от 0,89 до 0,78 см²/с. При переходе от диффузии по вакансиям (кривая 1) к диффузии по границам зерен металла (кривые 2, 3), где структура, как правило, чрезвычайно сильно искажена, процесс диффузии значительно ускоряется. Это связано с тем фактом, что при диффузии по границам зерен энергия активации диффузии хрома меньше, чем по вакансиям.

Заключение

В рамках вакансионного механизма залечивания трещин показано, что параметры диффузии микропор существенным образом влияют на скорость залечивания трещин в металле. Проведена оценка полного времени залечивания при различных температурах. Проведена оценка возможности использования вакансионного механизма залечивания трещин на примере диффузии атомов хрома по вакансиям (микропорам) к трещинам в α -Fe. Снижение на 25% коэффициента диффузии приводит к уменьшению скорости залечивания в 10 раз. Время полного залечивания при температурах 900–1000 К составляет 100...200 ч. Установлена зависимость скорости залечивания трещин от энергии активации диффузии атомов Cr. Полученные в работе результаты расчетов в рамках модели структурно-диффузионного механизма свидетельствуют о том, что скорость залечивания трещин главным образом определяется значением и изменением энергии активации диффузии атомов Cr. Помимо этого установлено, что увеличению скорости залечивания трещин способствует увеличение температуры отжига.

ЛИТЕРАТУРА

1. Завистовский, В.Э. Механика разрушения и прочность материалов с покрытиями / В.Э. Завистовский. – Новополоцк : ПГУ, 1999. – 144 с.
2. Физика твердого тела, уч. пособие для втузов / И.К. Верещагин [и др.] – М. : Высш. школа, 2001. – 237 с.
3. Zavistovskiy, V. On interaction between cracks and particles in coated materials / V. Zavistovskiy, E. Bogdanova, S. Zavistovskiy // Fracture mechanics and physics of construction materials and structures : Materials of II international symposium, 7–10.11.1996. – Lviv–Dubliany. – P. 45–48.
4. Фистуль, В.И. Физика и химия твердого тела : учебник для вузов : в 2-х т. / В.И. Фистуль – М. : Металлургия, 1995. – 486 с.

УДК 621.316.89

ПРОВОДЯЩИЙ КОМПОЗИЦИОННЫЙ МАТЕРИАЛ

А.Н. ЛАСТОЧКИН

(Представлено: канд. техн. наук, доц. Т.В. МОЛОДЕЧКИНА)

Проведено геометрическое моделирование свойств композиционных материалов. Установлено, что для обеспечения электропроводности диэлектрической матрицы необходимо ввести около 45% проводящего наполнителя.

Для получения элементной базы с улучшенными характеристиками необходимо использование материалов, обладающих конкретным набором физических, электрических и прочих свойств. Причем, в уже существующих материалах этот набор может отсутствовать или некоторые параметры могут не соответствовать предъявляемым требованиям. Успешное развитие современной электроники невозможно без разработки новых материалов. Поэтому перспективны композиционные материалы, структуру и состав которых можно подобрать для получения заданных свойств.

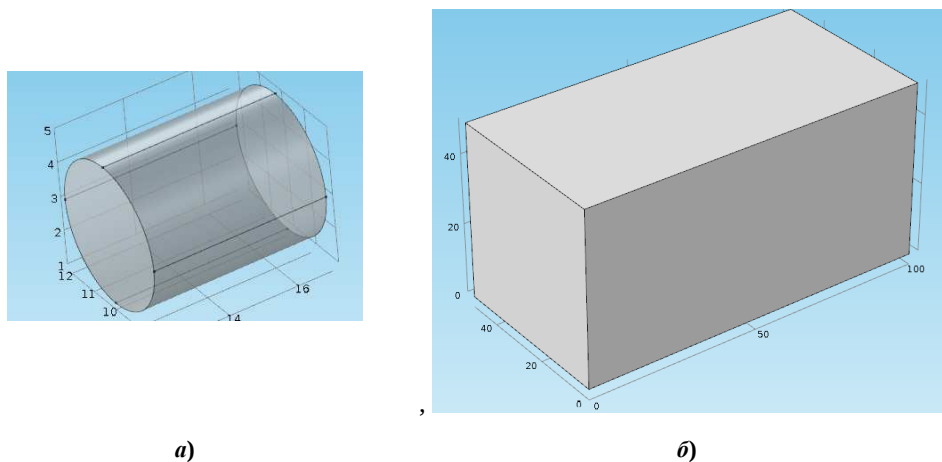
Композиты разрабатываются на основе матрицы, материала связывающего разнородные компоненты и позволяющего проводить технологические операции по изготовлению изделия заданной формы (монолит, пленка и т.п.). Матрица должна быть пластичной и обеспечивать высокую прочность контактного взаимодействия между компонентами, входящими в состав композита. Обеспечение теплофизических характеристик, устойчивости в условиях воздействия климатических факторов и различных излучений также относится к требованиям, предъявляемым к матрице. Используют в качестве матрицы металлические, полимерные, керамические материалы. Возможно использование глины, цемента [1].

Вещества-наполнители обеспечивают набор основных функциональных свойства композита. В качестве наполнителей выбирают компоненты, позволяющие повысить определенные свойства материала. По своей форме наполнители могут быть чешуйчатые, волокнистые, в виде порошков.

На практике для достижения необходимых свойств композита требуется изготовить большое число образцов материалов, в которых нужно комбинировать различные сочетания компонентов. Это связано с большими материальными затратами, длительностью процесса разработки состава композита.

Для ускорения работ по подбору состава композиционных материалов можно проводить моделирование свойств, используя для этого различные программные продукты. Разработаны программы, позволяющие проводить моделирование свойств с использованием математических или геометрических моделей. Нами опробованы такие программы, как ELCUT, COMSOL, Solid Works и т.д. [2] Но наиболее интересные результаты получены с использованием программы COMSOL, в которой для предсказания свойств композита использовали геометрическую модель, отражающую состав материала, размеры и взаимное расположение частиц наполнителя. Кроме того, в программе имеется возможность задания свойств матрицы и наполнителя, используя достаточно большую базу исходных материалов.

На рисунке 1 показана модель наполнителя и исходная модель композита.



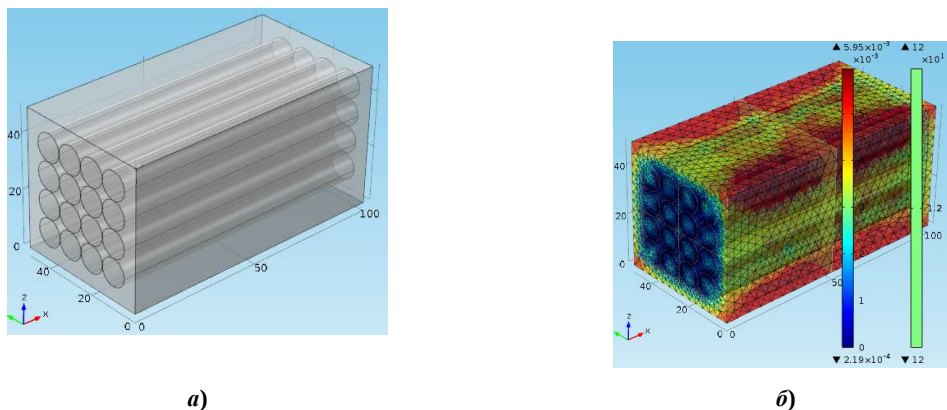
а – наполнитель; б – матрица

Рисунок 1. – Исходная модель композита

Геометрическая модель построена, наполнитель распределен необходимым образом, далее – заданы электрофизические свойства матрицы и наполнителя.

Моделирование проводилось для того, чтобы установить, какое минимальное количество вещества-наполнителя нужно использовать, чтобы добиться электропроводящих свойств диэлектрической матрицы.

На рисунке 2 показаны модель композита и результаты проведенного моделирования.



а – исходная модель композита; б – результат моделирование электропроводности

Рисунок 2. – Результаты моделирования

Заключение

В итоге проведенного моделирования мы могли менять концентрацию наполнителя в матрице и тем самым вычислить оптимальный вариант состава композита для достижения поставленных перед нами задач. Как показали проведенные исследования, для того чтобы композиционный материал перешел в проводящее состояние достаточно лишь 45% концентрации наполнителя от всего объема материала. Результаты моделирования будут использованы при разработке состава опытных образцов материала.

ЛИТЕРАТУРА

1. Евстифеев, В.В. Электротехнические материалы, пластмассы, смазывающе-охлаждающие жидкости, резины, композиты : учеб. пособие / В.В. Евстифеев, М.С. Корятов, В.Г. Азаров. – Омск : Изд-во СибАДИ, 2010. – 41 с.
2. Композиционные материалы / под ред. А.И. Монохина. – М. : Наука, 1981. – 292 с.
3. Волков, Г.М. Материаловедение : учебник / Г.М. Волков, В.М. Зуев. – М. : Издат. центр «Академия», 2008. – 400 с.

УДК 681.625.9

ПЕРСОНАЛЬНАЯ СТАНЦИЯ ПЕРЕРАБОТКИ ОФИСНОЙ МАКУЛАТУРЫ**Д.А. ГАВРИЛОВЕЦ****(Представлено: А.С. КИРИЕНКО)**

Исследуется проблема накопления офисной макулатуры и ее решение с помощью персонализированной технологии переработки такой макулатуры непосредственно в офисах. Раскрывается технология сухого разрыва бумажного волокна, дается экономическое обоснование целесообразности использования персональной станции переработки офисной макулатуры организациями, имеющими большие обороты бумаги.

В настоящее время организации, осуществляющие хозяйственную деятельность, сталкиваются с обработкой большого количества документации, хранением и утилизацией офисных бумаг [1]. Часто происходит огромное накопление черновиков и отработанных документов. Владельцы макулатуры вынуждены организовывать пункты хранения и выброса, заключать договора со спецавтобазами, предприятиями по переработке отходов или с третьими лицами сбора макулатуры, которые осуществляют ее вывоз. Это увеличивает трудоемкость процесса переработки и, как следствие, приводит к росту стоимости переработанного сырья.

Следовательно, разработка и применение современного персонального компактного оборудования для экологически чистой переработки макулатуры непосредственно на территории организации-пользователя в офисную бумагу является актуальной задачей, решение которой позволит снизить или полностью исключить вышеуказанные расходы [2]. Исследования в данной области соответствует приоритетному направлению фундаментальных и прикладных научных исследований Республики Беларусь на 2016–2020 годы (Перечень утвержден постановлением Совета Министров Республики Беларусь от 12.03.2015 № 190); многофункциональные материалы и технологии, а также направлению обеспечения ускоренного технологического развития отечественных отраслей экономики на основе создания и внедрения новых и высоких технологий для производства наукоемкой конкурентоспособной продукции Государственной программы «Наукоемкие технологии и техника» на 2016–2020 годы (Постановление Совета Министров Республики Беларусь от 21.04.2016 № 327).

Цель исследования – достижение решения научно-технической задачи переработки офисной макулатуры непосредственно в самом офисе, в котором должна быть установлена перерабатывающая мини-станция, решающая проблему накопления макулатуры.

Анализ проблемы переработки бумаги офисной макулатуры. Современное производство по переработке макулатуры имеет ряд технологий, позволяющих повторно использовать бумажные отходы. В настоящий момент уровень технологического обеспечения позволяет использовать 80% бумаги – нет возможности локальной её переработки. По причине растительного происхождения целлюлозы и ее волокнистой структуры при повторной обработке теряется свойство образовывать прочную связь между частью волокон.

На современных производствах используются такие виды технологий:

1) *гидроразрыв макулатурной массы.* Для данной технологии применяется огромная загрузка водных масс, что экономически нецелесообразно и требует утилизации отработанной водной массы с примесями металлов от красок и химикатов для очистки бумаги;

2) *физическая окраска бумаги без очистки макулатурной массы.* В данной технологии используется стандартная переработка в гидроразрывных машинах с последующей полимеризацией бумаги и ее окраски, поскольку не используются чистящие и отбеливающие средства;

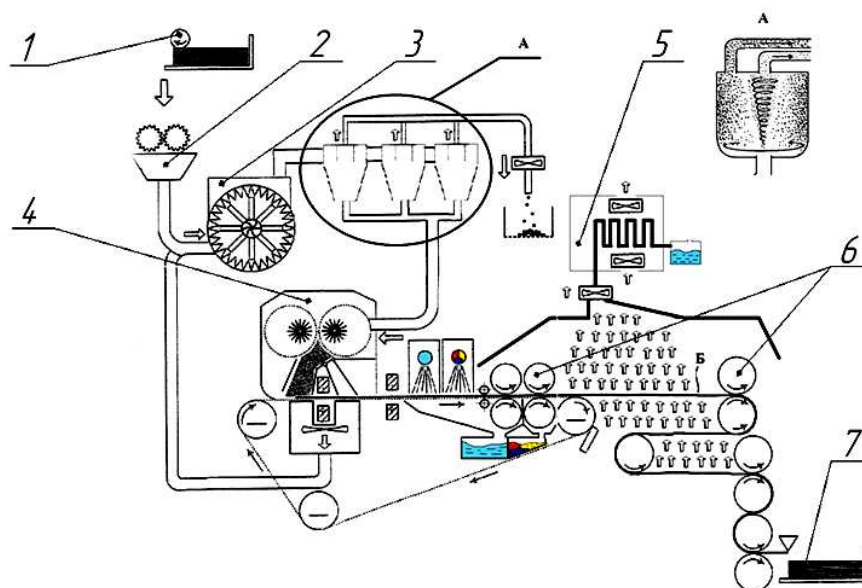
3) *технология с сухим разрывом макулатурного волокна.* Наиболее рациональная и экономически целесообразная технология по той причине, что в ней не используются химикаты для очистки отбелики суспензий и не требуются огромные объемы воды, которую в последующем надо переработать.

Описание предлагаемого решения. Предлагаемая конструкция перерабатывающей бумагу станции занимает малую площадь помещения в сравнении с огромными цехами по переработке макулатуры [3]. Для данной установки требуется подвод воды и электричества. Разработанная конструкция перерабатывает листы формата А4 с чернильными или красящими загрязнениями и формирует новые листы из переработанной массы макулатурного сырья. В процессе переработки в бумагу можно вводить различные материалы, изменяющие ее характеристики, такие как цвет, прочность, плотность и огнестойкость.

Предлагаемое решение технологии переработки офисной макулатуры имеет блочную последовательную систему, объединённую в одной компактной станции, которая разбита на блок-системы,

для удобства конструирования и разработки, а также последующего технологического обслуживания, замены расходных материалов, комплектующих, и ремонта.

Схема персональной станции переработки макулатуры представлена на рисунке.



- А – блок очистки бумажного волокна от загрязнений;
 1 – блок приема бумаги, в котором осуществляется сбор офисной макулатуры;
 2 – блок разрезания и измельчения бумаги,
 для измельчения бумаги до нужного размера при дальнейшей переработке;
 3 – блок механического разрыва бумажной массы;
 4 – блок формования, предназначенный для придания формы бумажному листу;
 5 – блок сушки и опреснения влаги; 6 – блок каландрирования – создания полимерных связей;
 7 – блок резки вывода бумажного листа

Схема персональной станции переработки офисной макулатуры

Конкурентные преимущества. Все существующие конкурентные технологии переработки макулатуры являются экологически грязными и трудоемкими, наносящими вред окружающей среде и экологии. В настоящее время в мире присутствует всего лишь один конкурент данной технологии и это компания Epson, производящая технику для печати.

Поэтому в качестве аналога к предлагаемому решению была взята запатентованная схема, компании Epson [3]. Основываясь на этой и других известных решениях по переработке макулатуры Китайских и Японских компаний, нами разработана технология и оборудование, которое по расчетным и экспериментальным данным увеличивает производительность и качество переработки в 2 раза.

Аналогичное производство на территории Республики Беларусь отсутствует, что является одним из положительных факторов развития использования данной инновации.

Заключение

В настоящее время в практическую организационную деятельность компаний все чаще внедряются электронные источники информации, организуются электронные базы документов и информационные офисы, но полный переход на них еще не близок. Поскольку до полного отказа от бумажных носителей информации еще далеко, предлагаемое нами решение вопроса экологически чистой персональной переработки офисной макулатуры на спроектированной станции является актуальной и экономически обоснованной задачей. Особенно это очевидно при огромных оборотах офисной макулатуры.

ЛИТЕРАТУРА

1. Фляте, Д.М. Технология бумаги : учебник для вузов / Д.М. Фляте. – М. : Лесная пром-сть, 1988. – 440 с.
2. Иванов, С.Н. Технология бумаги / С.Н. Иванов. – М. : Лесная пром-сть, 1960. – 712 с.
3. Способ получения суспензии целлюлозной массы бумажно-содержащего материала : пат. публикации № EP20020004578, 06.11.2002 ; пат. заявки № EP1254986 A1.02.05.2001 / Jürgen Dockal-Bauer, Lothar Dr. Pfalzer, Harald Selder. – Режим доступа: <https://www.google.com/patents/EP1254986A1?cl=ru>. – Дата доступа: 27.09.2017.

УДК 681.625.9

**СОВЕРШЕНСТВОВАНИЕ ЭФФЕКТИВНОСТИ БЛОКА ФОРМОВАНИЯ ПЛОТНОСТИ ЛИСТА
ПЕРСОНАЛЬНОЙ СТАНЦИИ ПЕРЕРАБОТКИ ОФИСНОЙ МАКУЛАТУРЫ****Д.А. ГАВРИЛОВЕЦ****(Представлено: А.С. КИРИЕНКО)**

Исследуется вопрос повышения эффективности блока формования плотности листа, применяющегося в персональной станции переработки офисной макулатуры. Выяснено, что решение этого вопроса позволит увеличить качество и наиболее точно создать бумажный лист, сформированный в персональной станции. Рассмотрены как положительные, так и отрицательные стороны данного блока формования, его устройство и технологии совершенствования эффективности.

Персональная станция по переработке макулатуры имеет ряд возможностей для усовершенствования, что повысит ее эффективность. В аналоге [2] блок формования не имеет реверса, регулировка в котором осуществляется по данным одного датчика, который измеряет толщину нанесенного слоя на сетчатую ленту. Изменение скорости подачи волокна на ленту осуществляется за счет увеличенной подачи макулатуры в разрывную машину. Такая система не имеет возможности изменения толщины слоя, пройденного через датчик контроля и имеет эффективность 75% поэтому разработка модифицированного блока позволит увеличить качество формования плотности бумаги. Предлагаемое усовершенствование имеет 2 датчика контроля и возможность как предварительного, так и окончательного регулирования толщины бумажного листа. Установленная дополнительная заслонка позволяет перенаправлять макулатурные массы из одной части камеры в другую и имеет эффективность от 80 до 95%.

Целью исследования является достижение решения научно-технической задачи усовершенствования блока формования персональной станции переработки офисной макулатуры, в котором эффективно создан технологический процесс контроля и нанесения макулатурного волокна на сетчатую ленту.

Описание предлагаемого решения. В патенте (рис. 1) принцип работы состоит в том, что волокно, очищенное от загрязнения, поступает в формовочную камеру, где попадает на сетчатые и игольчатые ролики 1 которые распутывают волокно и распределяют равномерно на ленте. Для лучшего удержания и формования волокна имеется сеточная лента, которая позволяет прилегать волокну к ней. Под сетчатой лентой установлен откачивающий насос 6, который способствует лучшему прилеганию волокна к ленте 4. Волокно движется по ленте и просвечивается датчиками контроля толщины 2, которые в случае уменьшения толщины замедляют конвейер и ускоряют разрыв бумажного волокна макулатуры. Форсунка 3 смачивает различными составами волокно для начала процесса полимеризации, после чего горячие ролики 5 предварительно сжимают и нагревают волокно, которое в последующем снимается с ленты.

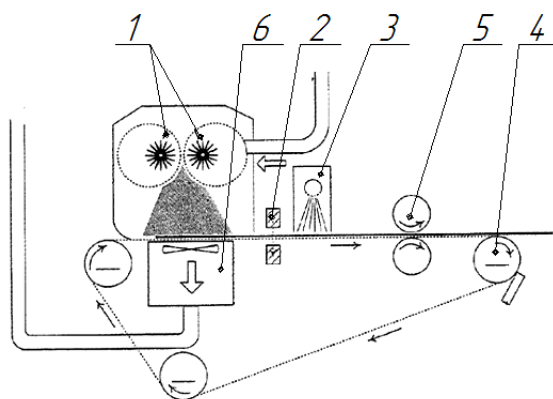


Рисунок 1. – Схема блока формования

Недостатками такой системы является контроль толщины волокна [3], который контролирует уменьшение толщины волокна, при этом ни как не локализирует толщину волокна, которое уже вышло из формовочной камеры. Решением будет техническое предложение на схеме (рис. 2), в котором введен дополнительный датчик контроля толщины 7 и дополнительной заслонки 9, которые будут регулировать толщину волокна предварительно при нанесении волокон на ленту и окончательно после ее регулировки.

Таким образом, наиболее лучший контроль за производимым бумажным волокном позволяет снизить возможность брака и снизить возможность увеличения затрат на переработку отбракованных листов бумаги.

Более развитая система смачивания [1] и окрашивания бумажного волокна позволит уменьшить затраты на воду. В которую входит 3 – две форсунки окраски и смачивания, 8 – предварительные ролики отжима волокна, которые позволяют снизить переизбыток водной массы на ленте в подсеточную ванну под роликами. Ролики 5 попарно не зависимы друг от друга и могут прижиматься поочередно различные виды бумаги. Попарно независимые ролики введены для окраски волокна перед формовкой различных видов бумаги, разделяя тем самым смачивающую камеру от окрашивающей.

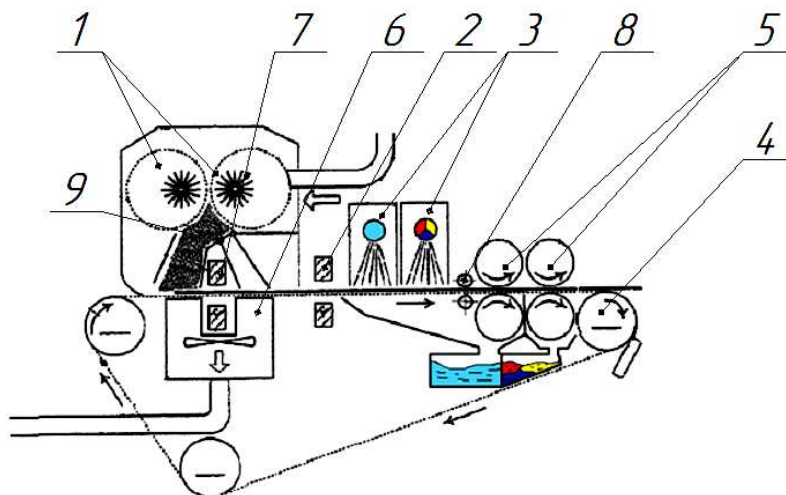


Рисунок 2. – Модифицированная схема блока формования

Конкурентные преимущества. Разработанный усовершенствованный блок формования наиболее эффективен по сравнению с аналогом компании Epson. Система нанесения макулатурного волокна на конвейере получила ряд дополнительных устройств, таких как установка дополнительной заслонки с датчиком контроля плотности, что позволило осуществлять контроль непосредственно в самой формовочной камере. Такое введение позволит по сравнению с конкурентами увеличить точность с существующих 70% в аналоге до 95% в предложенной модифицированной схеме.

Также в предложенной схеме присутствует предварительный ролик, который поможет сохранить и повторно использовать смачивающие материалы на 15% эффективнее конкурентов. Попарно независимые ролики увеличивают эффективность при выполнении формования, поскольку исключен процесс промывки роликов, то отсутствует время на промывку только одно пары нагревательных роликов как в аналоге компании Epson.

Заключение

Поскольку мировая тенденция складывается к уменьшению затрат на производство и увеличению прибыли, то формовочная камера наиболее эффективна и уменьшает затраты на повторную переработку брака, который может появляться в процессе нанесения макулатурной массы, смачивании и нагревании.

ЛИТЕРАТУРА

1. Иванов, С.Н. Технология бумаги / С.Н. Иванов. – 3-е изд. – М. : Лесная пром-сть, 2006. – 696 с.
2. Способ получения суспензии целлюлозной массы бумажно-содержащего материала : пат. публикации № EP20020004578, 06.11.2002 ; пат. заявки № EP1254986 A1.02.05.2001 / Jürgen Dockal-Bauer, Lothar Dr. Pfalzer, Harald Selder. – Режим доступа: <https://www.google.com/patents/EP1254986A1?cl=ru>. – Дата доступа: 27.09.2017.
3. Технология получения, обработки и переработки бумаги и картона: учеб. пособие для вузов / А.В. Вураско [и др.] ; под общ. ред. Н.Л. Медяник. – Екатеринбург : Урал. гос. лесотехн. ун-т, 2011. – 272 с.

УДК 629.33

**СОВРЕМЕННЫЕ МЕТОДЫ ЗАЩИТЫ
И ОКРАШИВАНИЕ КУЗОВОВ АВТОМОБИЛЕЙ****И.М. ЦЁМА***(Представлено: канд. техн. наук, доц. А.В. ДУДАН)*

Анализируются факторы окружающей среды, приводящие к утрате лакокрасочным покрытием автомобиля некоторых полезных свойств, от эстетических до защитных. Исследованы процессы коррозии кузова автомобиля и способы его замедления. Выявлено отличие покраски автомобиля на заводе-изготовителе и на станции технического обслуживания. Сделаны выводы о дальнейшем использовании автомобиля при повреждении кузова.

Кузовной ремонт недолговечен. Покраска автомобиля на заводе имеет существенное отличие от ремонта кузова на станции технического обслуживания (СТО).

Долговечность кузова определяет обычно срок жизни автомобиля, зависящий в основном от коррозионной стойкости металлических изделий и элементов. Обычно через 3–4 года эксплуатации легкового автомобиля на поверхности кузова появляются различные коррозионные повреждения, а через 5–6 лет эксплуатации начинается разрушение несущих элементов конструкции под действием коррозионных процессов.

Вредные воздействия на корпус автомобиля можно подразделить на три категории [1]:

- химические или электрохимические;
- механические;
- физические.

Основная задача покраски автомобиля – защита кузова от коррозии. Коррозия – главный враг автомобиля. Коррозия стали – процесс окисления металла кислородом в присутствии воды. На выходе получается гидратированный оксид железа – рыхлый порошок, который называется ржавчиной. Разрушения автомобильного кузова относят к классическим примерам электрохимической коррозии. Но вода и воздух – это лишь часть проблемы. Помимо обычных химических процессов важную роль в нем играют гальванические пары, возникающие между электрохимически неоднородными парами поверхностей. В автомобиле много мест, где образуются гальванические пары. Точки сварки, кузовные панели из разного металла, различные крепежные элементы и агрегаты, даже разные точки одной пластины с разной механической обработкой поверхности. Между ними всеми постоянно есть разность потенциалов, а значит, в присутствии электролита будет и коррозия.

Электрохимическая коррозия больше всего проявляется при атмосферной коррозии. Одним из основных факторов, определяющих скорость атмосферной коррозии, является влажность воздуха. Для чистой поверхности железа в условиях отсутствия загрязнений воздуха критическая влажность равна примерно 70%. При наличии на поверхности деталей пыли и грязи она снижается до 50%. Последнее объясняется тем, что мелкие твердые частицы служат центрами конденсации влаги, а крупные сами адсорбируют влагу. Дальнейшее увеличение влажности воздуха, а также повышение температуры увеличивают скорость атмосферной коррозии. Вот почему невысушенный автомобиль в теплом гараже корродирует быстрее. В плохо вентилируемых и обогреваемых гаражах коррозия кузовов автомобиля протекает быстрее, чем в необогреваемых и хорошо вентилируемых гаражах. При температурах ниже точки замерзания пленки влаги электрохимический процесс коррозии притормаживается. Колебания температуры во времени имеют также большое значение в связи с конденсацией и повторным испарением влаги на поверхности металла деталей. При небольших суточных перепадах температуры в закрытых профилях автомобиля конденсируется влага, причем конденсат практически не высыхает из-за недостаточной аэрации.

Атмосферная коррозия усиливается различными примесями, которыми почти всегда загрязнен воздух. Источниками таких загрязнений являются пыль в воздухе, грязь и химические средства, используемые для снижения обледенения на дорогах. Пыль проникает в закрытые полости, щели и зазоры и скапливается в них. При последующем увлажнении эта пыль образует коррозионно-активную среду. Грязь, прилипающая к днищу кузова автомобиля, даже в сухие периоды остается влажной, и коррозия продолжается за счет влаги, содержащейся в этой грязи. Распространенные химические средства против обледенения дорожных покрытий – хлориды натрия и кальция – являются также существенным фактором, способствующим коррозии автомобильных кузовов.

Механические воздействия – это удары твердых предметов (камней), приводящие к сколам покрытия, абразивное воздействие пыли и грязи (в частности, при мойке), царапины и сколы от прочих воздействий (в том числе и от ДТП). Физические воздействия – это влияние света и температуры. Ультра-

фиолетовый свет способен разрушать цветовые пигменты красок. Причем чем ярче цвет (синий, красный, желтый), тем более он подвергается такому воздействию (выцветанию). Колебания температуры, а также замерзание/таяние воды на поверхности кузова (как известно, с изменением объема) приводят также к разрушению структуры полимерной пленки и образованию микротрещин.

Современные заводские и ремонтные материалы достаточно прочны и позволяют надежно защитить кузов от сквозной коррозии. Например, сейчас производитель автомобиля дает гарантию на отсутствие сквозной коррозии на 10 лет. Некоторые виды кузовов, обычно на дорогих машинах, где используются специальные методы защиты (оцинковка, лужение, полиуретановые и битумные напыления, цинковые контактные грунты на сварных швах) могут продержаться без коррозии и более длительное время. Применение полного спектра защиты (все приемы) даст устойчивый эффект на 25–30 лет. Однако это экономически нецелесообразно, так как современные автомобили более подвержены моральному старению из-за быстрого развития техники. Однако не все потребительские качества кузова сопротивляются износу столь эффективно. Например, глянец поверхности может продержаться 1–2 года, в зависимости от качества дорог, способов мойки и т.д. Производитель дает гарантию на сохранение глянца порядка 1 года. С помощью специальных покрытий можно продлить этот срок в 1,5–2 раза. Однако впоследствии все равно потребуются полировка кузова с применением абразивных полиролей, что само по себе является фактором износа покрытия. Через определенный период (в среднем 5–10 полировок) потребуются повторное нанесение внешнего слоя покрытия для восстановления толщины пленки [2].

У станций техобслуживания нет возможности защитить кузов автомобиля от коррозии так, как это делают на производстве автомобилей.

В настоящее время заводы, выпускающие автомобили, в большинстве своем применяют цинкование и катодную защиту как основные способы защиты кузовов автомобилей от коррозии. Считается, что надежнее, но значительно дороже (отражается на себестоимости автомобиля), – это оцинкование металла автомобиля.

Оцинковка – один из самых мощных видов защиты кузова от электрохимической коррозии. Используют термическую оцинковку, гальваническую оцинковку и холодное цинкование. *Термическая оцинковка* заключается в полном погружении всего кузова перед сборкой автомобиля в специальный раствор. Этот метод применяется только крупными производителями европейских иномарок. Для этого используются большие технические емкости, которые наполнены цинкосодержащим составом. После погружения кузова состав прогревается до необходимой температуры, что способствует надежному закреплению частичек цинка на поверхности металла. В результате процедуры на поверхности металла создается тончайшая пленка толщиной до 2 мкм, которая препятствует проникновению влаги и губительному процессу окисления. Обработанные таким методом самые стойкие к коррозии автомобили показывают наилучшие результаты испытаний в соляных камерах.

Сам же производитель нередко дает поистине длительную гарантию на кузов. Иногда показатели достигают срока до 30 лет. Минимальный срок службы таких автомобилей составляет порядка 15 лет.

Оцинковку применяют на следующих автомобилях:

- **Audi.** Этот производитель считается пионером в области полной оцинковки кузова. Первый их автомобиль с оцинкованным кузовом был выпущен еще в начале 80-х годов. С тех пор Audi является мировым лидером в этой области, поэтому их авто с полностью оцинкованными кузовами считаются самыми долговечными. Они редко требуют серьезного кузовного ремонта, за исключением серьезных ДТП;

- **Porsche.** Для этого бренда качество всегда было на первом месте. И одним из основных способов подтвердить это качество компания считает полноценную обработку кузовов некоторых своих моделей термической оцинковкой. Первой машиной этой компании с полностью покрытым цинковым составом кузова была Порше 911;

- **Volkswagen.** Также уже много лет практикует термическую оцинковку всего кузова. По праву считается одной из самых долговечных марок среди машин среднего класса;

- **Seat.** Еще один из вариантов машин среднего класса, которые подвергаются полноценной оцинковке. При этом цена на их продукцию остается на вполне приемлемом уровне, что в совокупности выгодно выделяет ее среди конкурентов;

- некоторые модели Ford Sierra, Ford Escort, последние модели Chevrolet Lacetti и новые модели Opel Vectra и Astra.

Более простая в техническом плане обработка – оцинковка кузова машин гальваническим методом. Автомобили, обработанные таким методом, также имеют довольно длительную гарантию от производителя на предмет возникновения сквозной коррозии.

Оцинкованные гальваническим методом кузова имеют немного меньшую себестоимость, чем те, которые подвергались термической обработке. То же самое можно сказать и о долговечности защитного слоя – он служит намного меньше.

Процедура гальванического цинкования заключается в следующем:

- деталь или полностью весь кузов машины погружается в емкость с кислотным раствором цинка;

- к нему подсоединяется отрицательная клемма от источника постоянного тока;
- сама емкость подключается к положительной клемме.

Происходящий в емкости процесс (электролиз) способствует надежному закреплению растворенных частичек цинка на поверхности металла. В итоге получается образование надежного защитного слоя, который отталкивает влагу и предотвращает окисление металла. Этот метод немного дешевле предыдущего, поэтому и автомобили с такой обработкой несколько ниже в цене.

С такой обработкой выпускают автомобили марки Mercedes и BMW. Эти производители используют комбинированную защиту, которая включает в себя не только гальваническую оцинковку, но и подбор специальной стали. Помимо этого немецкие лидеры в этой области используют в качестве дополнительной защиты довольно внушительной толщины слой лакокрасочного покрытия. Такое решение позволяет им сделать себестоимость кузова на приемлемом уровне для получения прибыли. И в то же время такая защита является не менее эффективной, чем выработанные годами методы Audi.

Следует отметить, что гальваническая оцинковка, в отличие от термической, легко повреждается после небольшого ДТП. Поэтому после аварий какие-либо защитные свойства сразу практически пропадают. А вот термическая оцинковка частично может восстанавливаться на локальном уровне даже после аварий средней тяжести.

Метод холодного цинкования схож с методом гальванического цинкования, однако он более простой и менее дорогостоящий. Используется в основном для локальной обработки некоторых элементов кузова машин в домашних условиях. Выполняется такая оцинковка без погружения в емкость, цинкосодержащий раствор наносится на поверхность металла с помощью электрода, подключенного к положительной клемме источника питания. Сам металл подсоединяется к отрицательной клемме. Этот метод не применяется заводами изготовителями автомобилей.

Также существуют бренды автомобилей, которые используют лишь частичную обработку кузова своих машин, выдавая ее за полноценную оцинковку. Что касается отечественной автомобильной промышленности, то тут частичной обработке подвергаются последние модели Lada. К ним относятся Калина и Гранта, которые в последнее время, по утверждениям производителя, покрыты защитным слоем цинка почти на 40%. В этих машинах полностью подвергаются оцинковке только днище, пороги и арки.

При этом применяется только односторонняя оцинковка некоторых деталей кузова. Вторая сторона просто грунтуется и окрашивается традиционным способом. Такая защита рассчитана на то, что скрытая от глаз часть кузова будет надежно защищена от незаметного гниения.

Предложенная технология позволяет значительно сэкономить средства на производственных линиях и сделать хорошую рекламу своей продукции. К таким производителям также относятся Hyundai, Chery, Geely и Kia. При этом последний бренд нередко применяет обычное катафорезное грунтование с добавлением частичек цинка. Такая защита тоже весьма эффективна, однако она никак не может сравниться с обработкой, которой подвергаются автомобили европейских брендов.

Катафорезное покрытие – более дешевый, однако менее эффективный способ защиты кузова автомобиля от коррозии. Такое покрытие применяется на автомобилях ВАЗ. Технология сравнительно дешевая, но сложная. Для нанесения защитного покрытия, предохраняющего металл кузова от коррозии, автомобиль перед покраской проходит несколько стадий. Сначала металл, после сварочных работ, очищают от окислительной пленки (металл на воздухе моментально окисляется), затем обезжиривают и покрывают фосфатной пленкой. Впоследствии на кузов автомобиля наносят катафорезное покрытие. Это защитное покрытие можно нанести на металл кузова только в заводских условиях. На СТО таких технологий не используют.

Надежным методом защиты металла кузова от коррозии в заводских условиях может послужить электроосаждение грунта в специальной ванне. К ванне, заполненной разведенным автомобильным грунтом, подключается электрический провод (анод) со знаком «+». В эту ванну опускают целиком кузов автомобиля. К кузову подключают другой электрический провод (катод), ток со знаком «-». Под воздействием электрического поля происходит внедрение частиц грунта в поверхностный слой металлического кузова. При этом образуется прочная пленка грунта, которая проникает во все поры металла. Грунт как бы въедается в металл и равномерно покрывает всю поверхность детали автомобиля. Эта пленка и является надежной защитой металла кузова от коррозии. И только после этого кузов автомобиля грунтуют под покраску. Затем грунт сушат и красят автомобиль в нужный цвет. Это очень сложный процесс, состоящий из нескольких стадий:

- 1-я стадия – обезжиривание моющим составом;
- 2-я стадия – обезжиривание моющим составом с добавлением активатора;
- 3-я стадия – промывка водой;
- 4-я стадия – фосфатирование;
- 5-я стадия – струйная промывка в тоннеле;
- 6-я стадия – пассивация.

Сушка автомобиля осуществляется в тоннеле при температуре 200 °С в течение 30 минут, окраска производится эпоксидным грунтом и эмалью.

При ремонтных работах в условиях СТО невозможно осуществить такие стадии обработки кузова автомобиля, как цинкование, фосфатирование, пассивация, нанесение катафорезного грунта, которые обеспечивают основную коррозионную защиту кузова. В основном работа заключается в первичной подготовке поверхности (удаление загрязнений, мойка шампунями), обезжиривании (удаление органических загрязнений, битумных пятен, следов смазок), абразивном шлифовании (удаление старой краски, следов коррозии), создании антикоррозионного покрытия (первичное и вторичное грунтование) и нанесении эмали и лака [3]. Иногда при таких работах выполняется рихтовка механических повреждений кузова. Без выполнения электрохимического процесса защиты металла автомобиля, коррозия металла кузова наступит намного раньше.

В том месте, где повреждена заводская катафорезная или цинкованная пленка, обязательно образуется очаг коррозии, который через некоторое время поднимет слой шпаклевки и краски. Краска вспучивается, и коррозия вылезет наружу (жучки, карие, точки коррозии).

Если мастера на станции предлагают поменять элемент на новый, то нужно иметь в виду, что новый элемент не прослужит долгое время. Кузовные запчасти, поступающие в продажу, покрывают на заводе простым транспортировочным грунтом, который служит для защиты металла при перевозке, но совсем не защищает от воздействия коррозии. Покрашенный новый элемент (без заводской обработки) прослужит максимум 2–3 года и будет поражен коррозией насквозь.

Покрашенный элемент, обработанный катафорезным покрытием (заводским), выдерживает воздействие солевого тумана в камере от 1500 до 2000 часов. На точно таком же покрашенном элементе, но не имеющем катафорезное покрытие, сквозная коррозия в камере соленого тумана появляется уже через 90 часов. Поэтому любой ремонт кузова будет недолговечным. Более дорогой ремонт прослужит больше времени (за счет более точного выполнения технологии покраски), чем ремонт в собственном гараже.

Конечно, можно нанести катафорезное покрытие на ремонтируемый автомобиль или кузовную запчасть, но это потребует больших финансовых затрат. Целиком наносить катафорезную пленку перед покраской автомобиля выгодно только на раритетных автомобилях при их восстановлении.

Таким образом, можно сделать **вывод** о том, что если через год-два следы ремонта становятся видны, то вина мастера в этом небольшая. Любая станция, даже с самым современным оборудованием и высокопрофессиональными мастерами, не сможет повторить заводскую технологию покраски. Зная отличия в технологии покраски автомобиля на заводе и СТО, можно констатировать, что автомобиль, пострадавший в ДТП и требующий ремонта кузова, лучше сбывать не ремонтируя.

Если все же дорого отремонтировать автомобиль, покрасив его в ультрасовременной камере, целесообразней будет эксплуатировать его 2–3 года и сбывать до появления следов коррозии.

ЛИТЕРАТУРА

1. Альтхаус, Р. Автомобильные кузова: ремонт, уход, окраска / Р. Альтхаус. – М. : АСТ: Астрель, 2008. – 209 с.
2. Дамшен, К. Ремонт автомобильных кузовов [Электронный ресурс] / К. Дамшен. – М. : КЖИ «За рулем», 2004. – 234 с.
3. Технология покраски авто, или Как покрасить автомобиль самому [Электронный ресурс] / Автомобильный информационный портал // АвтоОмск. – 2005. – № 39 (358). – Режим доступа: http://autoservis555.narod.ru/krasim_avto.html/. – Дата доступа: 25.08.2017.

УДК 629.331

**АЛЬТЕРНАТИВНЫЕ МЕТОДЫ
ДИАГНОСТИРОВАНИЯ ДВИГАТЕЛЕЙ АВТОМОБИЛЕЙ****В.И. ХОМЧЕНКО, Е.А. ЖАВОРОНОК**
(Представлено: В.В. КОСТРИЦКИЙ)

Проведен сравнительный анализ эффективности, трудоемкости и результативности традиционных методов диагностирования двигателя и альтернативного метода под названием «Тест CSS». Выявлены преимущества и недостатки альтернативной методики диагностирования двигателя.

Для комплексного диагностирования бензинового двигателя на автотранспортных предприятиях с целью выявления неисправностей в кокой-либо из систем используются методики, основанные на применении классических средств диагностирования. Например, для оценки состояния цилиндропоршневой группы используется компрессометр, который измеряет пиковое давление в цилиндре при прокрутке двигателя стартером. Для проверки топливной системы используется топливный манометр, позволяющий оценить давление в системе, а также различные стенды для проверки форсунок с их снятием с двигателя, позволяющие оценить их производительность, качество распыла, эффективность. Для проверки системы зажигания используют: стенды для проверки свечей зажигания, тестеры сопротивления и т.д. При использовании этих методов диагностирования (в нашей работе будем их называть традиционные) в зависимости от марки и модели автомобиля тратится достаточно много времени, что существенно снижает показатели эффективности предприятия [1].

Однако за последние годы в сфере диагностирования двигателей предложены и нестандартные методики (в нашей работе будем их называть альтернативные), которые призваны повысить эффективность и снизить трудоемкость работ.

В данной работе мы рассмотрели и сравнили эффективность, трудоемкость и результативность традиционных методов диагностирования и альтернативного метода под названием «Тест CSS», разработанный диагностом Андреем Шульгиным для мотор-тестеров, произведенных в странах СНГ.

Тест CSS. Название этого теста происходит от первых букв немецкого названия датчика коленчатого вала – CrankShaftSensor. Сигнал датчика частоты вращения / положения коленчатого вала содержит большое количество информации о работе двигателя. В процессе работы каждый из цилиндров двигателя «подталкивает» коленчатый вал, за счет чего коленчатый вал кратковременно ускоряется после прохождения точек ВМТ 0° каждого из цилиндров. Если топливо в цилиндре не воспламенилось, происходит уже не ускорение, а замедление частоты вращения коленчатого вала. Даже тогда, когда блок управления двигателем непрерывно регулирует частоту вращения коленчатого вала при работе двигателя на холостом ходу с целью поддержания оборотов в заданном диапазоне, толчки от работающих цилиндров присутствуют, а от неработающих отсутствуют. Сигнал от датчика коленчатого вала совместно с сигналом о моменте искрообразования в 1-м цилиндре содержит в себе информацию о значительном количестве параметров двигателя [2].

Анализ этих сигналов позволяет:

- оценить статическую и динамическую компрессию для каждого из цилиндров;
- выявить неисправности в системе зажигания;
- оценить состояние топливных форсунок;
- получить характеристику подсистемы опережения зажигания;
- выявить биение задающего зубчатого диска;
- выявить пропущенные и согнутые зубья задающего зубчатого диска.

Для анализа скрипта CSS нужны сигнал от датчика коленчатого вала и сигнал синхронизации с моментом воспламенения в одном из цилиндров.

Сигнал частоты вращения удобнее всего снимать со штатного датчика коленчатого вала, а сигнал синхронизации – с высоковольтного провода одного из цилиндров для бензиновых двигателей, или с управляющего сигнала топливной форсунки для дизельных двигателей. В зависимости от особенностей конструкции двигателя и устройства его системы управления, могут применяться различные способы получения этих сигналов. Но следует заметить, что такие сигналы можно снять практически с любого двигателя, независимо от марки, модели и года выпуска [2].

В рамках данного теста диагностирование двигателя выполняется следующим образом. Мотор-тестер USB Autoscope необходимо подключить к USB-порту компьютера. Затем подать питание на мотор-тестер от диагностируемого автомобиля. Подключить к мотор-тестеру высоковольтный датчик синхронизации с искрой в системе зажигания и установить на высоковольтный провод первого цилиндра.

Затем подсоединить осциллографический щуп к входу № 5 USB Autoscope для снятия сигнала с датчика положения коленчатого вала (ДПКВ). Сигнальный щуп удобно подключить к сигнальному проводу ДПКВ при помощи иглы в разъем датчика. Открыть программу USB Oscilloscope и включить режим CSS.

Затем запускаем двигатель и оставляем работать его на холостом ходу. В окне программы включаем запись данных, и через несколько секунд поднимаем обороты до 3000 об/мин, затем резко закрываем дроссельную заслонку. Ждем холостого хода, после чего резко открываем дроссель. По достижении 4000 об/мин выключаем зажигание. При этом дроссельная заслонка должна быть открыта до полной остановки двигателя. Выключаем запись осциллограммы.

Анализ записанных осциллограмм. Для этого в окне программы выбираем пункт «выполнить скрипт». В результате выполнения анализа его результаты будут представлены в окне программы «USB Oscilloscope» в нескольких вкладках отчета: отчет, эффективность, опережение, зубчатый диск. Из четырех вкладок нас интересует вкладка «Эффективность» [3].

Вкладка «Эффективность». Во вкладке «Эффективность» отображаются график частоты вращения коленчатого вала двигателя и графики эффективности работы для каждого из цилиндров. Серый график отображает изменение оборотов двигателя за все время проведения замеров, а цветные графики – отдачу от каждого из цилиндров (рис. 1).

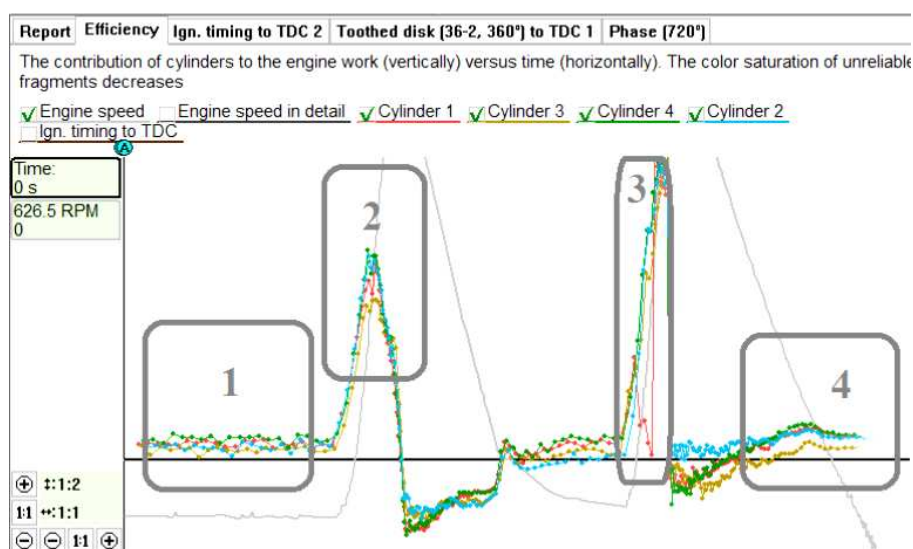


Рисунок 1. – Вкладка «Эффективность» из отчета скрипта CSS

Вкладка «Эффективность» является основной вкладкой отчета скрипта. Скрипт анализирует ускорение после воспламенения в каждом цилиндре, которое называется эффективностью цилиндра. Скрипт фиксирует эффективность цилиндра на протяжении всей работы двигателя и строит график эффективности цилиндра [3].

На эффективность работы цилиндра влияют 3 параметра: компрессия, надежное искрообразование и правильное соотношение компонентов топливно-воздушной смеси. По графикам эффективности можно выявить отклонения этих параметров и сделать предварительные выводы о состоянии механики двигателя, системе зажигания и системе топливоподдачи.

Различные фрагменты цветных графиков позволяют оценить для каждого из цилиндров:

- 1) холостой ход – этот фрагмент позволяет оценить стабильность работы двигателя на холостом ходу;
- 2) состав топливно-воздушной смеси – здесь проявляются загрязненные форсунки и подсосы воздуха во впускной коллектор;
- 3) бесперебойность искрообразования – позволяет выявить неисправности в системе зажигания;
- 4) динамическая компрессия – по этому фрагменту можно выявить износ поршневых колец, неплотные клапана газораспределения и ухудшенную наполняемость цилиндра воздухом.

Диагностирование двигателей автомобилей. Нами было проведено диагностирование двигателей различных марок автомобилей по традиционным методикам. Затем выполнили диагностирование по альтернативной методике с помощью теста CSS, сравнили затраченное время в первом и во втором случаях и эффективность результатов.

Первый автомобиль – это автомобиль Volkswagen Bora 1,6 AKL 1998 года. При проведении проверок по традиционным методикам не было выявлено неисправностей, а время на комплексную оценку

технического состояния двигателя составила 184 минут, так как для извлечения форсунок, свечей зажигания и для замера компрессии понадобилось демонтировать впускной коллектор.

При использовании теста CSS для комплексного диагностирования двигателя также не было выявлено неисправностей (рис. 2).

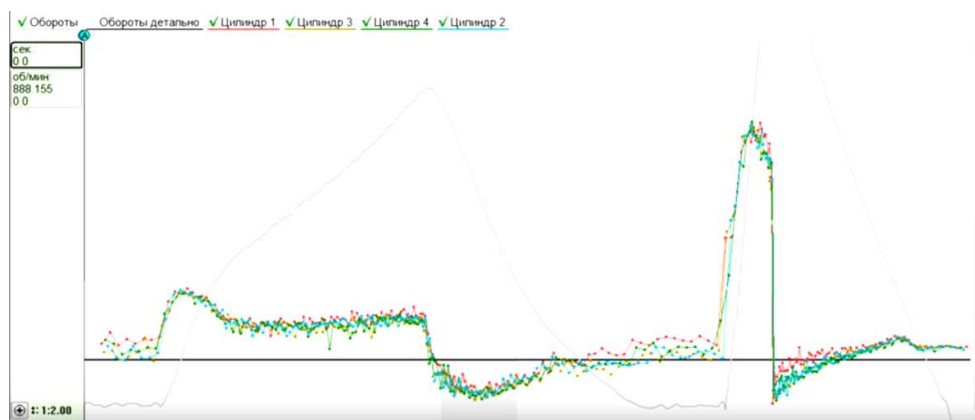


Рисунок 2. – Вкладка «Эффективность»

На рисунке 2 видно, что компрессия во всех цилиндрах одинаковая. Система зажигания работает исправно. С топливоподачей проблем тоже нет (все графики на всех режимах работы находятся на одном уровне). При этом время диагностирования составило 18 минут (в основном время было потрачено на поиск данного датчика).

Второй автомобиль – это автомобиль Fiat Marea 1,6 1996 г. В случае с этим автомобилем было конкретное проявление неисправности, которое проявлялось в виде неравномерной работы двигателя на всех режимах. Для этого автомобиля сначала был выполнен альтернативный тест, а затем проведены проверки по традиционным методикам. При этом подключение датчиков и методика аналогична. В этом автомобиле датчик коленчатого вала расположен таким образом, что для подключения к нему нам пришлось снять часть впускного тракта. Поэтому время на подключение и анализ результатов составило около 30 минут. При анализе замечено, что график эффективности 1-го цилиндра находится ниже остальных на всех режимах работы (рис. 3).

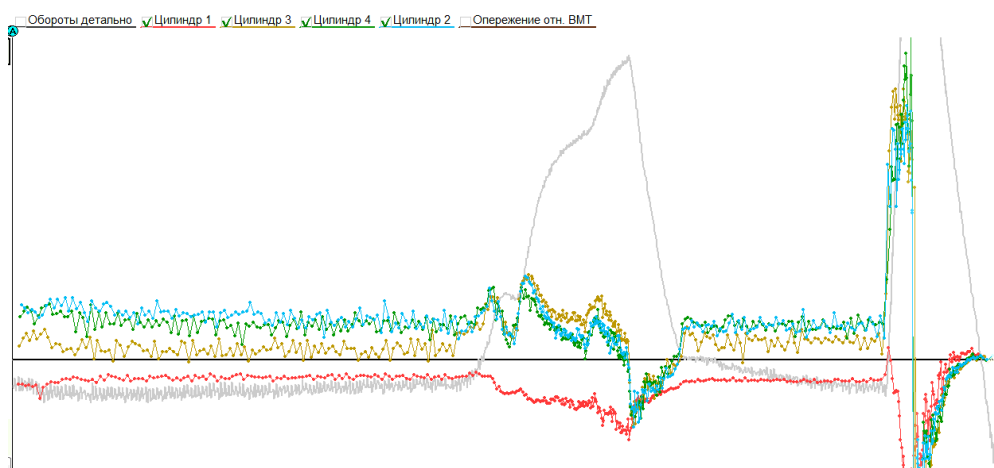


Рисунок 3. – Загрязнение топливной форсунки в 1-м цилиндре

Такое расположение графика свидетельствует о том, что форсунка не добавляет топлива, поэтому в цилиндр подается бедная смесь. Проблемы в цилиндропоршневой группе и в системе зажигания при помощи этого альтернативного теста выявлены не были.

Затем было проведено диагностирование основных систем двигателя автомобиля Fiat Marea с применением традиционных методов. Если демонтаж свеч, их проверка на стенде, проверка катушки высоковольтных проводов и измерение компрессии, а также определение давления в топливной системе не заняло много времени (44 мин), то проверка форсунок на стенде потребовала снятия впускного коллек-

тора и использования для этого подъемника. Поэтому время, затраченное на эту операцию, оказалось существенным (190 мин). Общее время составило 234 минуты.

При проверке элементов системы зажигания и оценке результатов измерения компрессии и давления в топливной системе не было выявлено неисправностей, как при первом тесте. Но при оценке работы форсунок на стенде было выявлено, что форсунка 1-го цилиндра имеет более низкие показатели производительности, чем другие форсунки на всех режимах работы. После очистки форсунок в ультразвуковой ванне результат остался прежним, поэтому было принято решение заменить форсунку на новую. После замены форсунки и установки ее на автомобиль повторно был произведен тест CSS, который указал на одинаковую эффективность всех цилиндров (рис. 4).

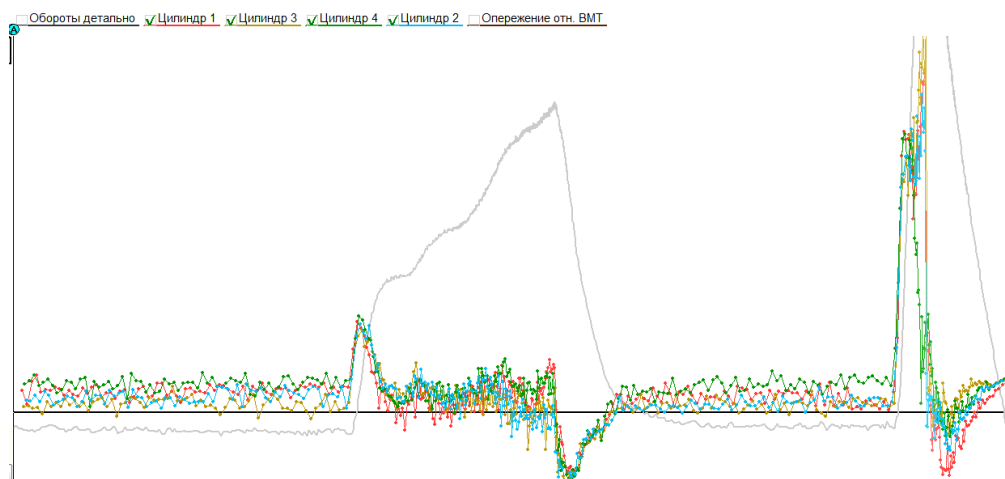


Рисунок 4. – Эффективность работы цилиндров автомобиля Fiat Marea после замены форсунки 1-го цилиндра

Проведем диагностирование еще одного автомобиля с целью определения конкретных неисправностей при помощи теста CSS для того, чтобы убедиться в его результативности. В данном случае – это автомобиль Mazda 626 2,0 16 v 2000 года. Этот автомобиль также имел признаки неисправностей, которые характеризовались как потеря мощности и нестабильная работа двигателя под нагрузкой. После проведения теста CSS, который занял около 15 минут, во вкладке «Эффективность» заметно отклонение красного и желтого графиков, отражающих эффективность работы первого и третьего цилиндра. По графику эффективности первого цилиндра видны пропуски воспламенения во время резкой перегазовки, что указывает на проблему с искрообразованием в первом цилиндре (рис. 5).



Рисунок 5. – Проблема с искрообразованием в 1-м цилиндре

Отклонение в конце графика эффективности третьего цилиндра указывает на проблему с компрессией, что стало причиной снижения мощности отдаваемого этим цилиндром (рис. 6).

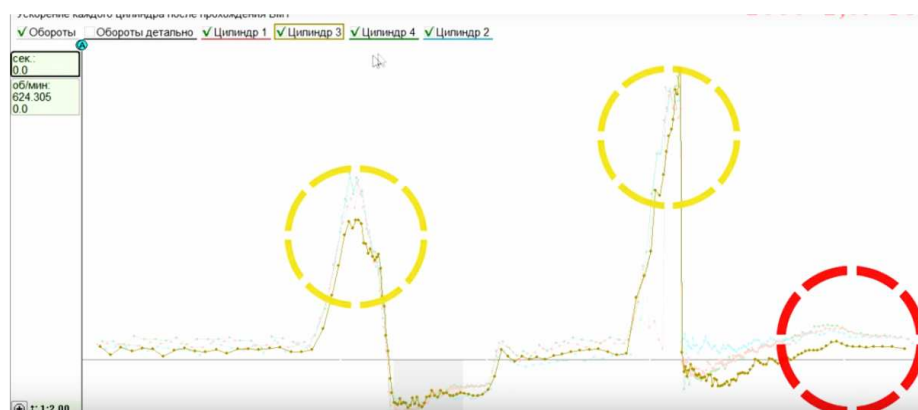


Рисунок 6. – Проблема с компрессией

Графики эффективности второго и четвертого цилиндров не содержат признаков неисправности. Как оказалось, пропуски воспламенения в первом цилиндре под нагрузкой возникли из-за пробитой по изолятору свечи зажигания и ее высоковольтного наконечника. Причиной снижения компрессии в третьем цилиндре был прогоревший клапан.

Заключение

Подытоживая результаты проведенного исследования можно сделать следующие **выводы**:

1. Альтернативный метод исследования при помощи скрипта CSS является незаслуженно игнорируемым в сфере диагностирования автомобиля.
2. Альтернативные методы диагностирования так же точны, как и традиционные, но имеют преимущество – требуют значительно меньших затрат времени.
3. Выполнение диагностики двигателя при помощи скрипта CSS достаточно простое, вместе с тем минусом, или сложностью, могут являться как проблемный доступ к датчику положения коленчатого вала, так и оптические датчики коленчатого вала, применяемые на некоторых автомобилях (например, Daewoo Matiz), с которых невозможно получить необходимый для теста CSS сигнал.

ЛИТЕРАТУРА

1. Савич, Е.Л. Техническая эксплуатация автомобилей : учеб. пособие : в 3 ч. / Е.Л. Савич. – Минск : Новое знание; М. : ИНФРА-М, 2015. – Ч. 2: Методы и средства диагностики и технического обслуживания автомобилей. – 364 с.
2. [Электронный ресурс]. – Режим доступа: http://chiptuner.ru/content/pub_33/. – Дата доступа: 04.05.2017.
3. [Электронный ресурс]. – Режим доступа: http://injectorservice.com.ua/docs/publications/css_typical_examples.pdf. – Дата доступа: 04.05.2017.

УДК 627.724

ВОССТАНОВЛЕНИЕ ШАРОВЫХ ОПОР ПРИ ПОМОЩИ ПОЛИАМИДА

В.И. ХОМЧЕНКО, Е. А. ЖАВОРОНОК

(Представлено: д-р техн. наук, проф. В.П. ИВАНОВ)

Настоящее исследование проведено с целью рассмотрения одного из возможных способов восстановления несъемных и неразборных шаровых опор при помощи полиамида, а также подбора материалов и технологического оборудования.

Введение. Шаровая опора пришла на смену конструкции, соединяющей цапфу колеса и рычаг подвески под названием «шкворень». Назначение шаровой опоры – сохранение колеса в горизонтальном положении во время вертикального перемещения, т.е. когда происходит его поворот. На рисунке 1. Показана конструкция шаровой опоры.

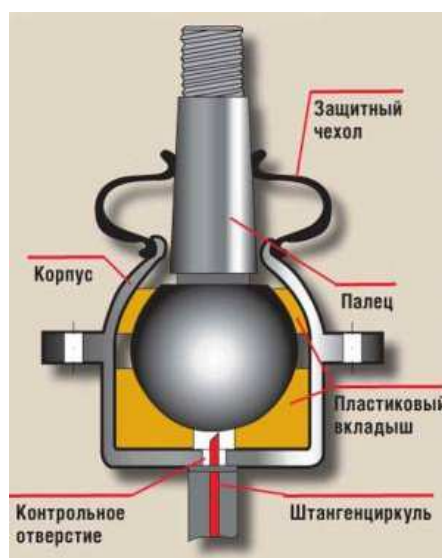


Рисунок 1. – Устройство шаровой опоры [4]

Конструкция шаровой опоры состоит из конусообразного вращающегося и перемещающегося на небольшие углы пальца, вставленного в корпус со сферическим или грибообразным наконечником. Важно помнить, что на эту деталь приходится основная масса автомобиля, поэтому необходимо уделять особое внимание шаровой опоре во время проведения очередного технического осмотра автомобиля. Эксплуатация автомобиля с разбитой шаровой опорой недопустима, так как при определенных положениях пальца и существующих нагрузках рычаг может отсоединиться от ступицы, с последующим отрывом колеса.

Обоснование восстановления или замены шаровой опоры

Со временем изнашиваясь, шаровые опоры начинают стучать, потому что в процессе эксплуатации испытывают серьезные нагрузки, так как в зависимости от места установки и конструкции подвески они могут нести на себе значительную часть массы автомобиля и выдерживают систематические удары при проезде неровностей. Основной причиной нарушения нормальной работы шаровой опоры является износ трущихся поверхностей, что приводит к увеличению зазора между корпусом и пальцем. В результате этого палец не только вращается, но и начинает перемещаться поступательно. При чрезмерном износе шаровой опоры ударные нагрузки способны привести к вырыванию пальца из корпуса. На вопрос, что делать с застучавшей шаровой опорой, можно дать однозначный ответ – менять на новую. Однако не всякую опору можно заменить отдельно от рычага. Есть опоры, которые крепятся к рычагам болтами или заклепками, и опоры, которые «вживлены» в рычаг, что при выходе опоры из строя предполагает замену рычага вместе с ней. Если стоимость отдельно меняющихся шаровых опор в зависимости от марки производителя, подразумевающей качество изделия, для распространенных в Беларуси легковых автомобилей укладывается в среднем в ценовой диапазон от 10 до 50 у.е., то стоимость рычага, не самого лучшего с точки зрения качества, гораздо дороже (50 у.е. и выше). Причем по прогнозной наработке

службы 30–100 тыс. км пробега шаровая опора не относится к самым долговечным узлам, что делает востребованной услугу восстановления опор и реставрации рычагов, практикуемую некоторыми СТО, специализирующимися на ремонте подвески.

Рассмотрим технологию восстановления шаровых опор при помощи полиамида. Диагностирование выявило предельный зазор между шаром и вкладышем опоры. Возникает дилемма – заменять или восстанавливать. На рисунке 2 в зонах, отмеченных красным цветом, износ максимален, также указаны места восстановления втулки полиамидом.

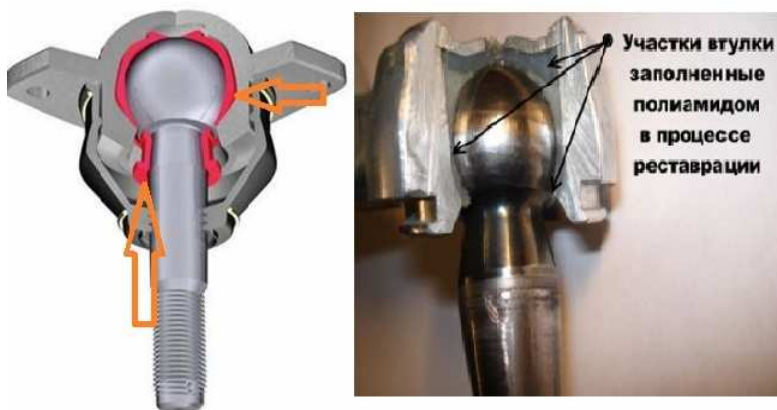


Рисунок 2. – Зона износа и последующего восстановления [5]

Видим, что материал однороден. Узел состоит из двух частей – обоймы и поворотного шара. Между ними находится полимер, который демпфирует жесткие удары и снижает трение между металлическими частями. Система работает по принципу суставной сумки в ноге человека. Именно этот полимер со временем изнашивается – появляется опасный люфт. Поскольку разобрать шаровую опору для замены вкладыша невозможно, остается одно – расплавить полимер и залить его внутрь.

Выбор полимеров для восстановления шаровой опоры

На наш взгляд, лучше всего выбирать из следующих полиамидов:

1) капролон (Полиамид-6 блочный; ПА-6 блочный). Капролон устойчив к воздействию углеводородов, масел, спиртов, кетонов, эфиров, щелочей и слабых кислот, растворяется в фенолах, концентрированных минеральных кислотах, муравьиной и уксусной кислотах. Материал имеет хорошие антифрикционные свойства и высокую износостойкость; при нормальных условиях не токсичен, не оказывает вредного воздействия на организм человека.

При механической обработке капролона в изделии разложения материала не происходит, вредные вещества не выделяются (капролон разлагается с выделением оксида углерода и аммиака при температуре выше 300 °С). Капролон имеет белый цвет с небольшой желтизной, без запаха. Его оттенок часто меняется благодаря различным красителям или присадкам, добавляемым в полимер в процессе производства.

2) TECAMID 66 (ТЕКАМИД 66) а также TECAMID 11 и TECAMID 12. Основными свойствами TECAMID 66 являются высокая стойкость к топливным материалам, маслам, жирам, большинству органических растворителей и щелочей. Одно из достоинств TECAMID 66 – высокая усталостная стойкость и высокая температуростойкость. Обладает высокой механической демпфирующей способностью.

Выбор материалов

Для восстановлени понадобятся следующие средства и материалы:

- 1) верстак с тисками для закрепления шаровой опоры или же шаровой опоры в сборе с рычагом;
- 2) сверлильный станок или же дрель со сверлом 5,0;
- 3) метчик метрический М6х1,0;
- 4) газовая горелка Пропан-Бутан;
- 5) масленка;
- 6) машина для литья под давление термопластичных материалов Термопластавтомат TRX-400;
- 7) переходник от аппарата к шаровой опоре с резьбой М6х1,00;
- 8) компрессор.

Порядок действий при восстановлении шаровых опор

1. Установив и надежно закрепив шаровую опору отдельно или же непосредственно со снятым рычагом, необходимо удалить с поверхности все видимые следы загрязнений. Затем снять пыльник и проверить его на наличие старения резины, разрушения и при необходимости заменить. После чего при помощи компрессора продуть место стыковки корпуса и пальца.

2. При помощи сверлильного станка или дрели со сверлом 5,0 просверливаем отверстие в дне шаровой опоры до шара, затем при помощи метчика метрического М6×1.00 нарезаем резьбу. Это необходимо для того, чтобы вкручивался переходник и в конце работ тавотница для защиты от попадания грязи и иных частиц.

3. В (аппарат) загружаем рассмотренные выше полиамиды и нагреваем их примерно до температуры 170–225 °С. Перемещаем шаровую опору или же рычаг в сборе и закрепляем при помощи механизма запираания форм. При помощи газовой горелки прогреваем шаровую опору для того, чтобы старый полиамидный вкладыш внутри шаровой опоры также расплавился.

4. Вворачиваем переходник по нарезанной резьбе и подсоединяем (аппарат). При достижении необходимой температуры под давлением примерно 0,20–0,22 МПа подаем расплавленный полиамид в шаровую опору. После заполнения при помощи компрессора шаровую опору охлаждаем и отсоединяем (аппарат), выкручиваем переходник.

5. После полного застывания в отверстии снова сверлят отверстие до шара, для того чтобы закрутить тавотницу и со временем шприцевать, когда в шаровой опоре притрется шар и вкладыш из полиамида. До закручивания тавотницы необходимо капнуть масла.

На этом восстановление шаровой опоры закончено и можно устанавливать ее на автомобиль.

Заключение

На основании сведений из литературных источников, а также мнений мастеров и клиентов, можно сделать вывод, что шаровая опора прослужит такой же срок, как и новая. Однако заметим, что восстановление обходится от 2-х и более раз дешевле, чем покупка новой шаровой опоры.

ЛИТЕРАТУРА

1. Флойд, Д.Е. Полиамиды / Д.Е. Флойд ; пер. с англ. Э.М. Левиной и М.Г. Гурария ; под ред. К.Н. Власовой. – М. : Гос. науч.-техн. изд-во хим. лит., 1960. – 180 с.
2. Беккер, М.Б. Литье под давлением / М.Б. Беккер, М.Л. Заславский, Ю.Ф. Игнатенко. – М. : Машиностроение, 1990. – 400 с.
3. Гордон, М.Дж. [мл.] Управление качеством литья под давлением / М.Дж. Гордон ; пер. с англ. – СПб. : Научные основы и технологии, 2012.
4. Устройство шаровой опоры [Электронный ресурс]. – Режим доступа: http://murdermorgan.blogspot.com.by/2012/12/blog-post_8570.html. – Дата доступа: 16.04.2017.
5. Зона износа и последующего восстановления [Электронный ресурс]. – Режим доступа: <http://auto-nm.ru/nashi-uslugi/remont-i-vosstanovlenie-sharovykh-opor>. – Дата доступа: 16.04.2017.

УДК 621.791

ВОССТАНОВЛЕНИЕ ДЕТАЛЕЙ
ПЛАСТИЧЕСКИМ ДЕФОРМИРОВАНИЕМ МАТЕРИАЛА

М.Ю. ЖЕРНОСЕК

(Представлено: д-р техн. наук, проф. В.П. ИВАНОВ)

Изучены существующие способы восстановления деталей поверхностным пластическим деформированием. Проанализированы их достоинства и недостатки. Получена зависимость относительной глубины пластической деформации от усилия прижатия обкаточного ролика и установлено, что увеличение толщины деформированного слоя происходит до значения усилия 2300 Н, а затем наблюдается отслоение поверхностного слоя материала.

Восстановление детали – комплекс технологических операций по устранению дефектов детали, обеспечивающих возобновление ее работоспособности и геометрических параметров, установленных нормативно-технической документацией.

Восстановление деталей давлением заключается в восстановлении первоначальных размеров рабочих поверхностей пластическим деформированием за счет перераспределения материала детали. В процессе деформирования материал детали вытесняется с нерабочих участков на изношенные поверхности, в результате чего восстанавливаются форма и размеры этих поверхностей.

Способность металлов к пластической деформации зависит от их пластических свойств, которые, в свою очередь зависят от химического состава, структуры, температура нагрева и скорости деформации. Чистые металлы имеют наибольшую пластичность, которая снижается с введением в их состав легирующих элементов [1].

Различают следующие виды пластической деформации [2] – холодная и горячая.

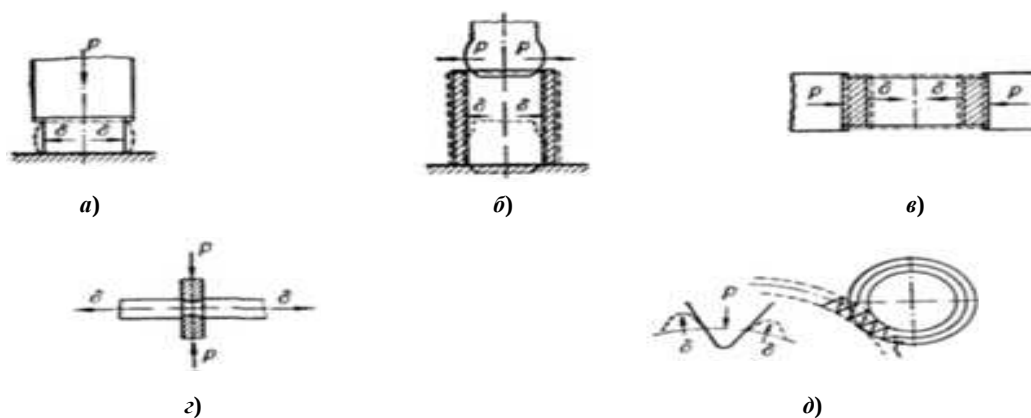
Первая осуществляется за счет приложения значительных внешних сил, сопровождается внутрикристаллическими сдвигами металла и его уплотнением. Этот вид деформации чаще все применяют при ремонте деталей из цветных металлов.

Второй вид деформации достигается предварительным подогревом детали до кованных температур. В этом случае происходят межкуристаллические сдвиги металла, требуется меньшая внешняя сила, поверхностного упрочнения металла не происходит.

Основные требования, предъявляемые к упрочняемым или восстанавливаемым деталям:

- наличие запаса материала на нерабочих участках ремонтируемой детали;
- достаточная пластичность материала;
- объемы механической и термической обработки должны быть минимальными;
- при ремонте этим способом закаленных или поверхностно-упрочненных деталей необходимо предварительно произвести отпуск или отжиг детали.

Процессы восстановления давлением изношенных деталей: осадка, раздача, обжатие, вытяжка, накатка представлены схематично на рисунке 1.



а – осадка; б – раздача; в – обжатие; г – вытяжка; д – накатка

Рисунок 1. – Виды обработки деталей давлением

Осадка (см. рис. 1, *а*) применяется для увеличения наружных размеров сплошных и полых деталей и уменьшения внутренних размеров полых. При осадке направление внешней силы P , действующей по вертикальной оси детали, не совпадает с направлением деформации.

Раздача (см. рис. 1, *б*) применяется для увеличения наружных размеров детали при сохранении или незначительном изменении ее высоты. В этом случае направление действующей силы P совпадает с направлением требуемой деформации, и металл перемещается от центра к периферии.

Обжатие (см. рис. 1, *в*) используется для уменьшения размера внутренней поверхности полых деталей за счет уменьшения размера ее наружной поверхности. При обжатии направление действующей силы совпадает; направлением требуемой деформации, происходит перемещение металла к периферии к центру.

Вытяжка (см. рис. 1, *г*) применяется для увеличения длины детали за счёт местного сужения ее поперечного сечения на небольшом участке. При вытяжке направление действующей силы P не совпадает с направлением требуемой деформации.

Накатка (см. рис. 1, *д*) применяется для увеличения наружных или уменьшения внутренних размеров детали за счет выдавливания металла на отдельных участках поверхностей. При накатке направление действующей силы P противоположно направлению требуемой деформации.

Преимуществами восстановления деталей пластическим деформированием является экономичность процесса, высокое качество, использование стандартного оборудования.

Недостатки – необходимость повторной термической обработки для некоторых случаев, потребность в специальной оснастке для каждого типоразмера.

В работе были проведены исследования зависимости толщины деформированного слоя от усилия прижатия ролика и установлено, что она носит нелинейный характер (рис. 2).

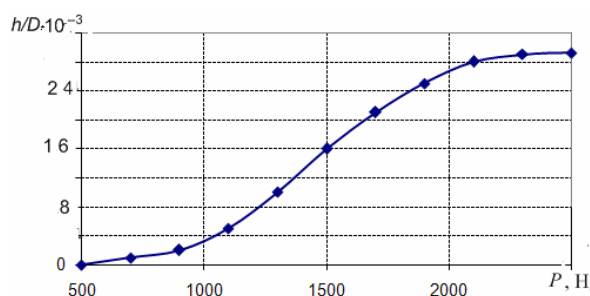


Рисунок 2. – Зависимость относительной глубины пластической деформации h/D от усилия прижатия обкаточного ролика P

Условно график можно разделить на три участка. На первом участке ($P = 500\text{--}1000$) Н, толщина деформированного слоя с увеличением усилия прижатия ролика изменяется незначительно, что объясняется преимущественно смятием микронеровностей. На втором участке $P = 1000\text{--}2000$ Н, зависимость носит примерно линейный характер. На третьем участке ($P > 2000$ Н), с увеличением усилия прижатия ролика толщина деформированного слоя практически не изменяется, так как верхний слой материала имеет ограниченную способность к аккумуляции энергии деформирования и по мере последовательной деформации наступает состояние энергетического насыщения. Дальнейшее увеличение усилия или количества ходов при поверхностной пластической обработке может привести к отслоению поверхности.

Заключение

В результате поверхностного пластического деформирования в поверхностном слое получена повышенная концентрация дефектов кристаллической решетки, которые тормозят плоскости скольжения, затрудняя их дальнейшее распространение, что позволяет повысить усталостную прочность валов.

Установлено, что толщина деформированного слоя увеличивается до определенного значения P (≈ 2300 Н), дальнейшее увеличение усилия прижатия обкаточного ролика не имеет смысла и зачастую приводит к отслаиванию материала.

ЛИТЕРАТУРА

1. Шадричев, В.А. Основы технологии автостроения и ремонт автомобилей : учебник для вузов / В.А. Шадричев. – Л. : Машиностроение, 1976. – 500 с.
2. Рохматулин, М.Д. Технология ремонта тепловозов : учебник для вузов / М.Д. Рохматулин. – М. : Транспорт, 1983. – 319 с.
3. Эксплуатация и ремонт машин и оборудования нефтяных и газовых промыслов [Электронный ресурс]. – 2017. – Режим доступа: <http://expl-remont.narod.ru/index/0-29>. – Дата доступа: 16.10.2017.

УДК 629.331

**ФУНКЦИОНАЛЬНЫЙ ТЮНИНГ АВТОМОБИЛЕЙ С ЦЕЛЬЮ ПОВЫШЕНИЯ
ТОПЛИВНОЙ ЭКОНОМИЧНОСТИ И УЛУЧШЕНИЯ ЭКОЛОГИЧЕСКИХ ПАРАМЕТРОВ****Е.А. ШЕВЧЁНОК, Е.Ю. ЯКИМОВИЧ**
(Представлено: В.В. КОСТРИЦКИЙ)

Ракрывается основная задача представленной работы – рассмотреть наиболее актуальные средства снижения расхода топлива и повышения экологичности, а также оценить реальную эффективность функционального тюнинга.

В последние годы в фокусе внимания автопроизводителей находятся уже не только требования повышения функциональных характеристик автомобиля и соответствия стандартам безопасности и нормам эмиссии, зачастую неразрывно связанные друг с другом. Дорожающее бензиновое топливо – еще один важный и явно выраженный фактор, влияющий на активизацию усилий по повышению экономичности производимых автомобилей. Топливная экономичность, эффективность, экологичность – три важнейшие взаимосвязанные цели, которые стоят сегодня при разработке лидирующих автомобильных технологий.

Однако эти технико-эксплуатационные свойства можно улучшить и на автомобилях прошлых лет при помощи предназначенных для этого устройств различных приспособлений, которые достаточно широко представлены на белорусском рынке. Такие конструктивные изменения автомобиля получили название функционального тюнинга.

В данной работе мы рассмотрим реальную эффективность приспособлений функционального тюнинга, представленных на рынке.

Vortex (Вихрь) – устройство (рис. 1) для улучшения гомогенизации топливовоздушной смеси. Гомогенизация – это более равномерное распределение одного вещества в другом методом дробления и перемешивания взаимно нерастворимых веществ. В нашем случае это бензин, дизель или газ и воздух. Устройство представляет собой цилиндр, внутри которого выполнены проточки под некоторым углом, в результате чего происходит завихрение воздушного потока, а не значительное сужение сечения дроссельного узла приводит к увеличению скорости воздушного потока. Как следствие, заряд воздуха уплотняется, а топливо дробится на более мелкие части и лучше перемешивается с воздухом. В итоге в цилиндр попадает более однородная топливовоздушная смесь с большим содержанием кислорода. Следствием такой обработки топливовоздушной смеси является повышение качества сгорания топлива. Экономия топлива может достигать от 7 до 15%. Прирост крутящего момента до 10%.



Рисунок 1. – Vortex

Энергосберегающие шины для автомобиля. Главное преимущество энергосберегающих шин – в меньшем выбросе загрязняющих веществ в атмосферу, которое достигается меньшим сопротивлением качению. Рассмотрим на примере: согласно экологическим нормам у современного автомобиля выброс вредных веществ в атмосферу должен составлять не более 120 граммов на один километр. Благодаря применению энергосберегающих шин можно сократить выброс CO₂ на 4–5 г/км. Используя энергосберегающие шины, экономия топлива составляет 200 граммов на сто километров пробега.

Экоактиватор – устройство (рис. 2) служит для предварительной подготовки топлива перед сжиганием. Принцип действия экоактиватора довольно прост (например, если потереть эбонитовой палочкой по шерстяной ткани, то появится магнитное поле, и палочка начнет притягивать к себе ворсинки шерсти, т.е. палочка положительный потенциал, а шерсть отрицательный). На молекулярном уровне происходит то же самое. Так как топливо – это жидкость, и она постоянно пребывает в движении: при транспортировке, перекачке, эксплуатации, молекулы так же трутся друг о друга. По законам физики противоположные знаки притягиваются, и молекулы срастаются в молекулярные сгустки или кластеры. В таком виде топливо и попадает в камеру сгорания. Далее молекулы, которые были на поверхности кластеров, сгорают, а те, что были внутри, проходят транзитом и вылетают в трубу, т.е. не производят никакой полезной работы. Назначение данного устройства – заставить сгореть то, что вылетает в трубу.

Когда такое топливо попадает в равнонаправленные магнитные поля экоактиватора, молекулярные сгустки начинают совершать колебательные движения с разворотами на 180 градусов. В итоге, после прохождения целого ряда таких полей топливные кластеры распадаются на отдельные молекулы, которые получают свободный доступ кислорода и сгорают более полно. А это, в свою очередь, позволяет как увеличить мощность двигателя, так и уменьшить расход топлива на 15–20%.



Рисунок 2. – Экоактиватор

Усилитель процесса горения позволяет повысить качество сгорания смеси. Этот прибор (рис. 3) создает электрический резонанс, вследствие чего горение смеси в камере сгорания переходит из ламинарного горения в резонансное. В обычном случае после того, как свеча зажигания выдала искру, воспламенение топливной смеси движется по кривой: по металлу вдоль стенок, через поршень, через противоположную стенку камеры сгорания и только потом попадает в середину объема. Таким образом, зачастую часть топлива сгорать просто не успевает. При подключении усилителя процесса горения смесь воспламеняется практически одновременно во всем объеме камеры сгорания. Естественно при этом происходит более полное качественное и равномерное горение. Усилитель процесса горения увеличивает мощность двигателя на 7–10% а также снижает уровень выброса вредных веществ в атмосферу как минимум на 15–20% при этом происходит и снижение расхода топлива на 10–15%. Устройство подходит для всех типов двигателя.

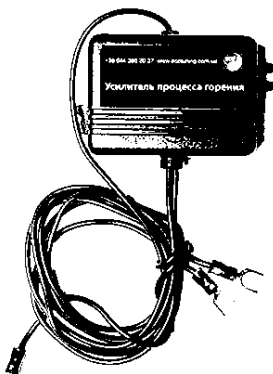


Рисунок 3. – Усилитель процесса горения

Аэродинамическое сопротивление автомобиля. Воздух обладает ощутимой плотностью: более килограмма на кубометр. Сила аэродинамического сопротивления увеличивается по квадрату скорости, собственно растет и расход топлива. На автомобиль с двигателем 1300–1500 см³ при снижении скорости со 130 до 120 км/ч расход топлива падает на 15%. Снижению расхода топлива способствует грамотный аэродинамический обвес. Низкий бампер и его юбка отсекает поток под днищем, накладки порогов сглаживают выступание колес, а спойлер организует поток – не позволяет возникать мощным тормозящим вихрям. Впрочем, заметного эффекта можно добиться и подгонкой кузовных деталей: щели и выступы могут увеличить расход топлива на скорости 100 км/ч до полулитра топлива на 100 километров.

Заключение

Представлено несколько вариантов, благодаря которым можно повысить топливную экономичность двигателя, которые доступны каждому автовладельцу. При использовании в комбинации на автомобиле таких устройств, как усилитель процесса горения и экоактиватора, суммарная стоимость составила 280 BYN. Однако результат стоит того. От автовладельцев, которые установили на свои автомобили совместно усилитель процесса горения и экоактиватор, мы получили следующие данные изменения расхода топлива их автомобилей:

- BMW 520 (2,5 л, 1996 г. в.) – замечено снижение расхода – 2 литра на 100 километров;
- Mitsubishi Pajero Sport (2,5 л, 2010 г. в.) – снижение расхода топлива – 3 литра на 100 километров;
- SuzukiGrandVitara (2,0 л, 2009 г. в.) – снижение расхода топлива – 3 литра на 100 километров;
- ВАЗ 2107 (2009 г. в.) – снижение расхода топлива – 3 литра на 100 километров.

ЛИТЕРАТУРА

1. [Электронный ресурс]. – Режим доступа: http://www.ecotuning.com.ua/activator_magnet.php. – Дата доступа: 04.06.2017.
2. [Электронный ресурс]. – Режим доступа: http://amastercar.ru/automaster/enorgosberegausehie_shiny.shtml. – Дата доступа: 04.06.2017.

УДК 629.331

**СРАВНИТЕЛЬНЫЙ АНАЛИЗ КАЧЕСТВА БЕНЗИНА
НА АЗС ГОРОДА НОВОПОЛОЦКА****Д.С. ШЕВЧЕНКО, В.В. ВОЙТЕХОВИЧ**
(Представлено: Л.И. ЛАГУН)

Исследуется качество бензина на АЗС города Новополоцка. Показано, что бензин на разных заправках существенно отличается по технико-эксплуатационным показателям топлива, которые оказывают существенно влияние на продолжительность эксплуатации транспортного средства, стабильность работы двигателя внутреннего сгорания и величину расхода топлива.

Применение некачественного бензина приводит к образованию паровых пробок, потере мощностных характеристик, перегреву двигателя, увеличению расхода горючего, а также к повышению нагара и смолистых отложений на деталях двигателя. При использовании бензина с высокой температурой конца кипения часть его поступает в цилиндры в капельно-жидком состоянии. Неиспарившаяся часть бензина по стенкам цилиндропоршневой группы стекает в масляный картер, разжижая моторное масло. При этом смазочные свойства масел резко ухудшаются, повышая износ деталей двигателя.

Применение бензинов, имеющих отклонения от их требований, ухудшает техническое состояние двигателя и приводит к повышенному выбросу токсичных веществ.

Топлива загрязняются на всех этапах своего жизненного пути – в процессе их производства, хранения, транспортирования, заправки техники и ее эксплуатации. Наличие в топливе загрязнений способствует образованию нагара на стенках камеры сгорания, поршнях и клапанах. Вследствие плохой теплопроводности нагароотложений температура стенок повышается, что ведет к увеличению концентрации оксидов азота. Нагар увеличивает объем «мертвого» пространства в камере сгорания, в результате чего увеличивается количество несгоревших углеводородов (примерно на 5–10% через каждые 10 тыс. км пробега автомобиля) [4].

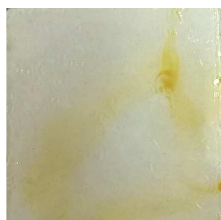
Исходя из вышесказанного, нами проведен сравнительный анализ качества бензина на заправочных станциях города Новополоцка: «Блок», «Белоруснефть», «Лукойл». Образцы реализуемого автомобильного бензина АИ-92 закуплены АЗС 26.04.2017.

Первый этап исследования – визуальный осмотр образцов на прозрачность. Образец АЗС «Блок» имеет слегка выраженный светло-бежевый оттенок, прозрачен. Образец АЗС «Белоруснефть» – более темный, прозрачен. Образец АЗС «Лукойл» – светлый, прозрачен.

Исследовались образцы бензина на наличие в них воды. Для этого использовался перманганат калия (далее – марганцовка). При попадании марганцовки в воду вода окрашивается в малиновый цвет. Во всех образцах марганцовка выпала в осадок и цвет самих образцов не изменился, можно сделать вывод о том, что во всех трех образцах вода отсутствует.

Проведен опыт на бумажном фильтре, который увлажнялся топливом и подвергался сушке. Чистое топливо испаряется полностью и не оставляет следов, а некачественное топливо может оставлять пятна от различных присадок и добавок. Образцы АЗС «Блок» и АЗС «Лукойл» не оставили после себя следов на бумаге, а образец «Белоруснефть» оставил небольшое пятно масляного цвета.

Немаловажным является исследование образцов после сгорания. Чистое и качественное топливо не оставит после себя следов горения и выгорит полностью. На небольшие стеклянные квадратной формы пластинки наливался соответствующий образец топлива и поджигался. Результаты сгорания топлива представлены на рисунке.



АЗС «Блок»



АЗС «Белоруснефть»



АЗС «Лукойл»

Результаты сгорания исследуемого топлива

Как видно из рисунка образцы с АЗС «Блок» и АЗС «Лукойл» оставили после себя незначительные пятна, образец с АЗС «Белоруснефть» оставил большое количество продуктов сгорания.

Октановое число бензина АИ-92 по исследовательскому методу должно равняться 92. Для определения октанового числа образцов был задействован прибор Октан-ИМ.

Необходимое условие для определения октанового числа трех образцов бензина АИ-92 – проведение установки нуля прибора. Для этого использовался н-гептан, октановое число которого по определению равно 0.

Далее произведены замеры октанового числа всех трех исследуемых образцов.

Октановое число образца АЗС «Блок» составило 86,8.

Октановое число образца АЗС «Белоруснефть» – 87,2, что не намного больше, чем на АЗС «Блок».

Октановое число образца АЗС «Лукойл» – 94,1.

Производителями трех образцов в день реализации являлись:

- на АЗС «Блок» и АЗС «Белоруснефть» – ОАО «Нафтан» г. Новополоцк;

- АЗС «Лукойл» – топливо, произведенное на территории Российской Федерации.

Заключение

На основании проведенного исследования и полученных результатов можно констатировать, что наиболее качественным образцом по визуальному контролю и величине октанового числа является бензин марки АИ-92 на АЗС «Лукойл».

Не всегда дорогой бензин означает то, что он качественный. Использование низкачественного топлива чревато выходом из строя топливной системы, существенно уменьшает ресурс двигателя и является причиной нестабильности работы, тяга уменьшается, запуск двигателя осложняется.

ЛИТЕРАТУРА

1. Кириченко, Н.Б. Автомобильные эксплуатационные материалы : учеб. пособие / Н.Б. Кириченко. – 2-е изд., стереотип. – М. : Академия, 2005. – 205 с.
2. Трофименко, И.Л. Автомобильные эксплуатационные материалы : учеб. пособие / И.Л. Трофименко, Н.А. Коваленко, В.П. Лобах. – Минск : Новое знание, 2008. – 231 с.
3. Васильева, Л.С. Автомобильные эксплуатационные материалы : учебник для вузов / Л.С. Васильева. – М. : Наука-Пресс, 2003. – 421 с.
4. Виноградов, О.В. Влияние показателей качества автомобильного бензина и дизельного топлива на состояние окружающей среды [Электронный ресурс] / О.В. Виноградов, А.С. Карелина // Молодой ученый. – 2016. – № 8. – С. 194–199. – Режим доступа: <https://moluch.ru>, свободный. – Дата доступа: 29.09.2017.

УДК 629.331

АНАЛИЗ СЕТИ АЗС В БЕЛАРУСИ

Д.С. ШЕВЧЕНКО, В.В. ВОЙТЕХОВИЧ
(Представлено: Л.И. ЛАГУН)

Анализируются сети автозаправочных станций Республики Беларусь, среди которых пять крупнейших в Беларуси: АЗС Белоруснефть; Лукойл; А-100; Газпромнефть; United Company. Выявлены факторы, проанализировав которые, АЗС можно ранжировать.

Качественное ли в Беларуси топливо? Все топливо соответствует стандартам. Следовательно, качество топлива не будет влиять на место в рейтинге лучших АЗС сетей в Беларуси.

Далее – стоимость топлива. Здесь ситуация уже интереснее. Всем известно, что цены на топливо устанавливает концерн «Белнефтехим». Для удобства зарубежных гостей ежедневно эти цены конвертируются в российские рубли, доллары и евро по курсу Нацбанка Республики Беларусь. Это означает, что цены на топливо в Беларуси фиксированные, только цены на газ указываются минимальные, а в реальности на заправках можно увидеть совершенно иные. Кроме этого, существуют многочисленные акции, скидочные и накопительные карты, купоны.

В сети АЗС «Белоруснефть» действует система скидок, при которой покупатель накапливает баллы, понижая таким образом стоимость следующих литров топлива.

В сети АЗС «Газпромнефть» действует программа лояльности «Разам з намі», участники которой при предъявлении дисконтной карты получают скидку от 2%.

Кроме качества топлива и стоимости, каждому водителю хотелось бы, чтобы при заправке был хороший сервис. На сайте компании «Лукойл» можно найти интересующую информацию об определенной АЗС. В таком случае водитель заранее будет знать, что его может ожидать от заправочной станции.

Автозаправочные станции ЛУКОЙЛ Белоруссия реализуют бензины и дизельное топливо стандарта ЕВРО-5, газовое топливо производства белорусских нефтеперерабатывающих предприятий (Мозырский НПЗ, Нафтан) и российских нефтеперерабатывающих заводов Группы «ЛУКОЙЛ» (ЛУКОЙЛ-Нижегороднефтеоргсинтез, ЛУКОЙЛ-Пермнефтеоргсинтез, ЛУКОЙЛ-Волгограднефтепереработка) [1].

На сайте компании «А-100» представлен широкий перечень дополнительных услуг, приятных моментов хорошего сервиса, однако не указано, на какой заправке это присутствует, а на какой – нет. У А-100 свой собственный склад нефтепродуктов. Именно там с помощью специального оборудования следят за качеством топлива. Перед сливом топлива в резервуары хранения берутся контрольные пробы для исследования в независимых аккредитованных лабораториях [3].

В сети АЗС «Белоруснефть» существует интересное предложение оплаты дорог в системе BelToll с помощью топливной карты. Такого предложения нет больше ни у одной сети, это сразу же выгодно отличает «Белоруснефть» на фоне конкурентов. Через фирменную сеть АЗС компании «Белоруснефть» реализуется моторное топливо производства белорусских нефтеперерабатывающих предприятий – ОАО «Мозырский НПЗ», ОАО «Нафтан» и газоперерабатывающего завода «Белоруснефти» [2].

АЗС сеть «Газпромнефть» даже не старается удержать обычных клиентов. Скорее всего это связано с тем, что компания свободно работает с оптовыми и корпоративными клиентами. Эта сеть славится своими акциями, креативным подходом к работе (например, акция на 1-е апреля, когда клиентов на АЗС обслуживали белорусские звезды). Но для среднестатистического клиента особенно хорошей эта сеть не покажется на фоне всех представленных выше.

Сеть АЗС «United Company» давно хорошо зарекомендовала себя на международном рынке. С сервисом этой сети автозаправочных станций сложно потягаться кому-либо. Интерфейс сайта компании расскажет, где конкретно вы сможете найти интересующие вас услуги (сами при этом выбираете набор нужных услуг). В отличие от компании «А-100», этой клиенты доверяют, хотя, в отличие от других, здесь нет скидочных карт, акций, купонов, отдельных станций, где какое-либо топливо стоит дешевле. За счет высококачественного сервиса можно не тратить на привлечение клиентов и удешевление самого топлива.

Лучшая АЗС сеть Беларуси по мнению Transportal – это ...

В номинации «Самая экономичная АЗС сеть» выигрывает «А-100» со всеми акциями, бонусами, выделенными заправками, купонами и картами. В номинации «Самая особенная АЗС сеть» побеждает «Белоруснефть», потому что только здесь существует услуга по оплате дорог. Победа в номинации «Самая

комфортная АЗС сеть» достается компании «Лукойл», которая стала золотой серединой. Тут есть и заправки с удешевленным и самым распространенным бензином, удобный сайт, большое количество заправок с сервисом на оценку «хорошо с плюсом».

В номинации «Самая благотворительная АЗС сеть» побеждает «Газпромнефть», которая вкладывает в это доброе дело силы, финансы, время.

Победителем среди победителей, самой-самой АЗС сетью становится компания «UnitedCompany» за постоянный порядок, чистоту, высококачественный сервис, пусть и без снижения цен на топливо.

Можно отметить, что каждая сеть пытается по-своему привлечь и удержать клиентов. Каждый водитель должен решить сам, где ему заправляться [4].

ЛИТЕРАТУРА

1. ИООО «ЛУКОЙЛ Белоруссия» [Электронный ресурс]. – Режим доступа: <http://lukoil.by>, свободный. – Дата доступа: 29.09.2017.
2. ПО «Белоруснефть» [Электронный ресурс]. – Режим доступа: <http://www.belorusneft.by>, свободный. – Дата доступа: 29.09.2017.
3. Сеть автозаправочных станций А-100 – официальный сайт [Электронный ресурс]. – Режим доступа: <https://azs.a-100.by>, свободный. – Дата доступа: 29.09.2017.
4. Транспортно-логический портал [Электронный ресурс]. – Режим доступа: <http://www.transportal.by>, свободный. – Дата доступа: 29.09.2017.

УДК 551.510.42

ОЦЕНКА ВОЗДЕЙСТВИЯ ВЫБРОСОВ АВТОМОБИЛЬНОГО ТРАНСПОРТА НА КАЧЕСТВО АТМОСФЕРНОГО ВОЗДУХА В ГОРОДЕ НОВОПОЛОЦКЕ

А.Л. ИВАНОВ**(Представлено: Л.И. ЛАГУН)**

Рассматривается проблема загрязнения атмосферного воздуха автотранспортом, которая с каждым годом становится все актуальней. Показаны основные источники загрязнения – объекты топливно-энергетической промышленности и автомобильного транспорта.

Уровень загрязнения воздуха в городе Новополоцке считается высоким. Такой уровень в первую очередь создает ОАО «Нафтан» завод «Полимир». Поэтому надо максимально уменьшить загрязнение воздуха от автомобильного транспорта. Город Новополоцк входит в список городов с самым высоким уровнем загрязнения.

Цель данной работы – оценка воздействия выбросов автомобильного транспорта на качество атмосферного воздуха на участке автодорог по ул. Молодежная и ул. Калинина.

Выбор участков определен высокой интенсивностью транспортного потока, а также разнообразием транспортных средств (ТС).

На сегодняшний день самым распространенным видом ТС является автомобильный транспорт. Количество единиц транспортной техники на улицах города Новополоцка непрерывно растет.

Качество городского атмосферного воздуха – основной фактор формирования санитарного и экологического благополучия городской среды.

При оценке работы двигателей внутреннего сгорания автотранспортных средств принято рассматривать следующие виды источников: открытая (закрытая) стоянка транспортных средств; работа техники; проезд ТС; обкатка двигателей ТС [3].

Для оценки состояния атмосферного воздуха проведены натурные наблюдения за потоком непрерывно движущихся ТС на участке автомобильных дорог по ул. Молодежная и ул. Калинина в городе Новополоцке.

Исследуемые участки показаны на рисунках 1 и 2.

При этом все ТС распределялись на грузовые, автобусы и легковые.

Полученные данные были обработаны при помощи статистических методов, что позволило получить усредненные данные по количеству автотранспортных средств, проезжающих на данном участке автодороги.

Также произведены замеры по улицам Молодежная и Калинина. Замер представлял собой, подсчет количества автотранспорта за промежуток времени 5 минут при 3 повторениях в 12:00 и 17:00.



Рисунок 1. – Улица Молодёжная



Рисунок 2. – Улица Калинина

Заключение

На основании результатов проведенного исследования и анализа общего количество движущегося автотранспорта в течение дня получены концентрации окиси углерода, которые сравнили с ПДК выбросов автотранспорта по окиси углерода, равной 5 мг/м^3 , сделан вывод – выбросы превышают ПДК [2]:

- ул. Молодёжная – днем – в 15,5 раз, вечером – в 18,9 раз;
- ул. Калинина – днем – в 8,7 раз, вечером – в 9,3 раз.

ЛИТЕРАТУРА

1. Транспорт и окружающая среда : учебник / М.М. Болбас [и др.]. – Минск : Технопринт, 2004. – 261 с.
2. Экология и ресурсосбережение на транспорте : учебник / М.М. Болбас [и др.] ; под ред. М.М. Болбаса. – Минск : Адукацыя і выхаванне, 2011. – 295 с.
3. Луканин, В.Н. Промышленно-транспортная экология : учебник для вузов / В.Н. Луканин, Ю.В. Трофименко. – М. : Высш. шк., 2001. – 296 с.

УДК 551.510.42

**ПРОБЛЕМА ЗАГРЯЗНЕНИЯ АТМОСФЕРНОГО ВОЗДУХА
ВЫХЛОПНЫМИ ГАЗАМИ АВТОТРАНСПОРТА В РЕСПУБЛИКЕ БЕЛАРУСЬ****А.Л. ИВАНОВ****(Представлено: Л.И. ЛАГУН)**

Рассматривается проблема загрязнения атмосферного воздуха выхлопными газами автотранспорта в Республике Беларусь, которая с каждым годом становится все актуальней. Показаны источники загрязнения, среди них и транспортные средства, вредные вещества при эксплуатации которых поступают в воздух с отработавшими газами, испарениями из топливных систем, при заправке. Приведено сравнение концентраций вредных веществ городов Беларуси и других стран.

Атмосферный воздух – это источник дыхания человека, животных и растительности; сырье для процессов горения и синтеза химических веществ; а также среда, в которую выбрасываются отходы жизнедеятельности человека, высших и низших животных и растений.

С каждым годом проблема загрязнения атмосферного воздуха автотранспортом становится все актуальней. Основными источниками загрязнения являются объекты топливно-энергетической промышленности и автомобильного транспорта.

Количество автомобилей в стране становится все больше, а с ними увеличивается и количество выбросов в атмосферу. В городах Беларуси 70% загрязнения воздуха приходится на автомобильный транспорт – к такому выводу пришли экологи, анализируя состав воздуха городов. В Минске на долю транспорта приходится 80%, всего доля автомобильных выбросов по республике составляет 65%.

Отходящие газы двигателей содержат сложную смесь, их более чем двухсот компонентов, среди которых немало канцерогенов. Наземные транспортные средства – это механизмы, передвигающиеся по шоссейным и железным дорогам, а также строительное, сельскохозяйственное и военное оборудование. В соответствии с различиями в количествах и видах выбрасываемых загрязняющих веществ целесообразно рассматривать в отдельности двигатели внутреннего сгорания (особенно двух- и четырехтактные) и дизели, аналогичным образом – паровые и дизельные локомотивы.

Вредные вещества при эксплуатации подвижных транспортных средств поступают в воздух с отработавшими газами, испарениями из топливных систем и при заправке, а также с картерными газами. На выбросы оксида углерода существенное влияние оказывает рельеф дороги и режим движения автомашины. Так, например, при ускорении и торможении в отработавших газах увеличивается содержание оксида углерода почти в 8 раз. Минимальное количество оксида углерода выделяется при равномерной скорости автомобиля 60 км/ч.

В связи с тем, что отработавшие газы автомобилей поступают в нижний слой атмосферы, а процесс их рассеяния значительно отличается от процесса рассеяния высоких стационарных источников, вредные вещества находятся практически в зоне дыхания человека. Поэтому автомобильный транспорт следует отнести к категории наиболее опасных источников загрязнения атмосферного воздуха вблизи автомагистралей.

Например, содержание мелкодисперсных частиц РМ10 в Минске низкое. В декабре 2016 года оно было 7–29 мкг/м³. Для сравнения: в воздухе китайского города Чунцин этот показатель достигал 612 мкг/м³, в Шанхае – 61, в Анкаре – 75, в Лондоне – 33, в Москве – 27, почти как в Минске.

Среднее содержание оксидов углерода в белорусских городах также не превышает 1 мг/м³. Это приблизительно уровень выбросов Лондона и Мадрида. В Москве и Париже он поднимается до 2 мг/м³. В некоторых районах Варшавы до 7 мг/м³.

Столица, город Минск, выбрасывает в воздух небольшое количество диоксида серы. Среднесуточный показатель равняется в среднем – 3,6 мг/м³, как в Мадриде. А вот в Гродно, Новополоцке и Могилеве содержание этого элемента выше – до 40–80 мг/м³. А это уже уровень промышленных городов Китая и Турции.

Согласно [1], мелкодисперсные частицы РМ10 – это пылинки, зола и другие частицы размером до 10 мкм. Высокое содержание их в воздухе способствует развитию заболеваний дыхательной системы.

Оксид углерода (СО) – газ, который выделяется при горении. Имеет ядовитые свойства, большая концентрация его в воздухе может вызвать головокружение и обморок, а более высокие концентрации – смерть.

Диоксид серы (SO_2) – газ, который выделяется при горении серы и сульфидов. Очень токсичен. При вдыхании вызывает насморк, кашель, охриплость. Более высокая концентрация вызывает удушье, рвоту, возможен острый отек легких.

По мнению сотрудников аналитического отдела мониторинга воздуха Белгидромета, по сравнению с азиатскими промышленными центрами населенные пункты Беларуси выгодно отличаются качеством атмосферного воздуха. Но и Беларусь имеет проблемные районы, особенно в промышленных городах: в Гомеле наблюдается превышение уровня оксида углерода, диоксида азота – в некоторых районах Могилева и Минска, мелкодисперсной пыли – в некоторых районах Минска, Гомеля и Могилева. А при повышенных температурах почти во всех городах повышается уровень формальдегида.

Длительное вдыхание автомобильных выхлопов ведет к общему снижению иммунитета и нарушению работы сердечно-сосудистой системы, провоцирует развитие болезней дыхательной системы.

Исследование, проведенное несколько лет назад учеными из Университета Южной Калифорнии, показало, что выхлопные газы также наносят значительный вред головному мозгу, что в дальнейшем может вылиться в развитие болезни Альцгеймера, а у детей даже до года вызывать аутизм.

Согласно [2], выбросы в атмосферу от автотранспорта снизились в Беларуси за последние семь лет на 350 тыс. тонн, по информации отдела регулирования воздействия на атмосферный воздух и озоновый слой Минприроды Беларуси. А выбросы загрязняющих веществ от автомобилей в атмосферу Беларуси в 2016 году сократились на 5% по сравнению с 2015 годом. Если в 2009 году среднереспубликанский выброс был 1 млн. 137 тыс. тонн, то в 2016 году он составил всего 791 тыс. тонн. То есть снизился выброс суммарно по республике за счет транспорта на 350 тыс. тонн. Несмотря на сокращение выбросов в атмосферу от автотранспорта, его доля в общем объеме загрязнения остается подавляющей. В Беларуси с 2013 года работает стратегия снижения вредного воздействия от транспорта до 2020 года.

По словам председателя Минского городского комитета природных ресурсов и охраны окружающей среды [3], одним из мероприятий по снижению вредного воздействия на окружающую среду от транспорта является Европейская неделя мобильности, которая проходит ежегодно с 16 по 22 сентября. По результатам проведения этой недели в Беларуси в 2015 году количество выбросов сократилось на 100 тыс. тонн, в 2016 году – на 170 тыс. тонн.

Заключение

В связи с тем, что отработавшие газы автомобилей поступают в нижний слой атмосферы, а процесс их рассеяния значительно отличается от процесса рассеяния высоких стационарных источников, вредные вещества находятся практически в зоне дыхания человека. Поэтому автомобильный транспорт следует отнести к категории наиболее опасных источников загрязнения атмосферного воздуха вблизи автомагистралей. Необходимо реализовывать мероприятия, подобные Европейской неделе мобильности, и другие по снижению вредного воздействия на окружающую среду от транспорта.

ЛИТЕРАТУРА

1. Экология и ресурсосбережение на транспорте : учебник / М.М. Болбас [и др.] ; под ред. М.М. Болбаса. – Минск : Адукацыя і выхаванне, 2011. – 295 с.
2. Минприроды: количество автомобильных выбросов за семь лет значительно сократилось [Электронный ресурс]. – Режим доступа: <http://naviny.by>, свободный. – Дата доступа: 30.09.2017.
3. Выброс загрязняющих веществ от автомобилей в атмосферу Беларуси в 2016 году сократился на 5%: [Электронный ресурс]. – Режим доступа: <https://thinktanks.by>, свободный. – Дата доступа: 30.09.2017.

ХИМИЧЕСКИЕ ТЕХНОЛОГИИ

УДК 665.6/7

ИССЛЕДОВАНИЕ СВОЙСТВ СЕРО- И ФОСФОРСОДЕРЖАЩИХ ПРИСАДОК К МОТОРНЫМ МАСЛАМ С УЛУЧШЕННЫМИ ХИММОТОЛОГИЧЕСКИМИ СВОЙСТВАМИ

Р.М. РЯБУХИН, Е.Я. КОРБУТ**(Представлено: канд. хим. наук, доц. С.В. ПОКРОВСКАЯ)**

Рассмотрены свойства диалкилдитиофосфатов цинка, а именно изменение их антиокислительных и противоизносных свойств в зависимости от использования различных компонентов в процессе синтеза (спиртов). Разработана методика синтеза новой для предприятия СООО «ЛЛК-НАФТАН» дитиофосфорной присадки RM-22. Показано, что новая присадка RM-22 обладает улучшенными трибологическими и антиокислительными свойствами.

Разработка присадок к маслам является следствием решения проблем, возникающих в результате прогресса двигателестроения на фоне ужесточения требований к экологической чистоте топлив и смазочных материалов.

На сегодняшний день моторные масла должны обладать широким спектром свойств, которые диктуются как со стороны потребителей и экологов, так и со стороны производителей оборудования. Сложившуюся непростую ситуацию может решить применение специальных веществ – присадок, которые в свою очередь могут придать маслу новые свойства, улучшить имеющиеся и замедлить нежелательные процессы [1].

Для улучшения смазывающей способности масел к ним добавляются присадки. Присадки, применяемые для улучшения смазочных свойств масел, по характеру действия разделяют на противоизносные и противозадирные.

Противоизносные присадки предотвращают интенсивный износ трущихся поверхностей при нормальных режимах трения без заедания [2].

В связи с тем, что уровень требований к смазочным материалам, а именно к моторным маслам, повышается с каждым годом и выпускаемые присадки в данный момент неспособны удовлетворять определенным параметрам, улучшение свойств диалкилдитиофосфатов цинка осуществлялось посредством использования в процессе синтеза других компонентов, а именно спиртов иного строения. Исходя из литературных данных и в результате проведенного маркетингового исследования, было решено использовать спирт производства PRASOL CHEMICALS LTD. под торговым названием MISCA, то есть метилзобутилкарбинол (2-гидрокси-4-метилпентан).

Предметом исследования являлись свойства диалкилдитиофосфатов цинка, а именно изменение антиокислительных и противоизносных свойств в зависимости от использования различных компонентов в процессе синтеза (спиртов).

С целью максимизировать практическую ориентированность проводимой работы, синтез новой для производства присадки проводился в условиях, которые воспроизводимы на производстве и имеют место при производстве таких присадок, как ДФ-11к, Н-102 и т.д. Основой методики послужил ряд патентов [6], [7], [8] и производственный регламент [9].

Полученная в результате синтеза присадка RM-22 была подвергнута следующим анализам качества:

1. Анализ на содержание элементов при помощи рентгенофлуоресцентного анализатора.
2. Анализ антиокислительных свойств: при помощи дифференциального сканирующего калориметра определялся индукционный период масла.
3. Анализ трибологических свойств: при помощи четырехшариковой машины трения определялся диаметр пятна износа.

Исходя из результатов рентгенофлуоресцентного анализа, можно утверждать, что большее количество фосфора, цинка, серы содержится в присадке ДФ-11к, что объясняется разницей молекулярных масс заместителей, входящих в состав этих присадок.

Из результатов анализа антиокислительных свойств присадок можно сделать вывод, что присадки в значительной степени увеличивают индукционный период, что положительно сказывается на свойствах масла. В большей степени положительным эффектом на индукционный период обладает присадка RM-22, причем заметное преимущество наблюдается и при самой малой концентрации, которая применялась в данной работе.

По результатам анализа размер диаметра пятна износа у масла с вовлечением присадки RM-22 меньше, чем у масла с аналогичным массовым вовлечением присадки ДФ-11к, что свидетельствует о большей эффективности противоизносных свойств присадки RM-22.

Сравнение трибологических и антиокислительных свойств новой присадки RM-22 со свойствами присадки ДФ-11к представлены в таблице 1.

Таблица 1. – Сравнение трибологических и антиокислительных свойств новой присадки RM-22 со свойствами присадки ДФ-11к

Параметр	Диаметр пятна износа, мм			Индукционный период, мин		
	0,8% масс.	1,0% масс.	0,1% масс. Zn	0,8% масс.	1,0% масс.	0,1% масс. Zn
Вовлечение присадки						
ДФ-11к	0,5	0,48	0,475	7,13	8,37	9,08
RM-22	0,38	0,37	0,36	8,56	9,61	10,15
Преимущество RM-22, мм	–0,12	–0,11	–0,115	+1,43	+1,24	+1,07
Преимущество RM-22, %	24,00	22,92	24,21	20,06	14,81	11,78
Среднее преимущество, %	23,71			15,55		

На основе полученных результатов важно сделать вывод, что новая присадка RM-22 обладает улучшенными свойствами, а именно:

1) полученная присадка RM-22 придает лучшие трибологические свойства смазочному материалу, а именно меньшее пятно износа на 23,7%, чем присадка ДФ-11к;

2) полученная присадка RM-22 придает лучшие антиокислительные свойства смазочному материалу, а именно больший индукционный период на 15,5%, чем присадка ДФ-11к.

Заключение

Совершенствование бензиновых и дизельных двигателей влияет на расширение ассортимента высокоэффективных присадок, на изменение структуры их потребления для обеспечения ужесточаемых требований увеличения сроков смены, повышения топливной экономичности. Рост потребности в высококачественных моторных маслах будет невозможен без наличия на рынке присадок и пакетов присадок, способных обеспечить требуемый уровень качества моторных масел. Поэтому в настоящее время большое внимание со стороны производителей масел уделяется повышению качества присадок. При выполнении научных исследований разработана технология получения противоизносной присадки с улучшенными трибологическими и антиокислительными свойствами.

ЛИТЕРАТУРА

1. Кулиев, А.М. Химия и технология присадок к маслам и топливам / А.М. Кулиев. – 2-е изд., перераб. – Л. : Химия, 1985. – 312 с.
2. Рудник, Л.Р. Присадки к смазочным материалам. Свойства и применение / Л.Р. Рудник ; пер. с англ. – 2-го изд. ; под ред. А.М. Данилова. – СПб. : ЦОП «Профессия», 2013. – 928 с.
3. Kurt Pollak, Esso Research and Engineering company : Pat. No. 3.546.324.
4. Harold Shaub. United States : Pat. 4.105.571.
5. United States Patent 3.234.250 Patented F oh. 8. 1966. Preparation of ZDDP / Helmuth G. Schnider, James Clark.
6. Технологический регламент цеха № 1 по производству алкилфенольных присадок. – Новополоцк : СООО «ЛЛК-НАФТАН», 2009.

УДК 657.6:66

ЭКОНОМИЧЕСКАЯ ЭФФЕКТИВНОСТЬ ВНЕДРЕНИЯ НОВЫХ ТЕХНОЛОГИЙ ПРОИЗВОДСТВА СЕРО- И ФОСФОРСОДЕРЖАЩИХ ПРИСАДОК К МОТОРНЫМ МАСЛАМ С УЛУЧШЕННЫМИ ХИММОТОЛОГИЧЕСКИМИ СВОЙСТВАМИ

Р.М. РЯБУХИН, Е.Я. КОРБУТ**(Представлено: канд. хим. наук, доц. С.В. ПОКРОВСКАЯ)**

Рассмотрена конкурентоспособность новой противоизносной присадки на рынке как самостоятельный продукт. Разработана технология синтеза новой для предприятия СООО «ЛЛК-НАФТАН» дитиофосфорной присадки RM-22. Показано, что с экономической точки зрения отпускная стоимость новой присадки ниже, чем отпускная стоимость зарубежных аналогов.

Применение смазочных масел, легированных современными присадками, обуславливает длительную и безотказную работу энергетических установок и оборудования. Качество масла может быть улучшено при введении одних присадок и совершенно не изменится при введении других того же функционального действия, поэтому присадки относятся к наукоемкой продукции, характеризующейся тем, что затраты на разработку сопоставимы с затратами на организацию их промышленного производства [1].

Противоизносные присадки предотвращают интенсивный износ трущихся поверхностей при нормальных режимах трения без заедания. В условиях умеренных нагрузок и температур противоизносными присадками могут служить многие ПАВ. Однако при трении соприкасающиеся поверхности значительно нагреваются и адсорбционная способность смазки уменьшается. Поэтому в качестве противоизносных присадок применяют лишь те ПАВ, которые при повышении температуры способны реагировать с поверхностями металла и образовывать пленки, препятствующие схватыванию поверхностей. Такими веществами являются некоторые соединения, содержащие неактивную серу, а также эфиры кислот фосфора [2].

Конкурентоспособные присадки должны иметь приемлемую растворимость в базовом масле, быть не очень дорогостоящими и не слишком сильно снижать окислительную стабильность масел или усиливать коррозию контактирующих с ним металлов. Ранее широко применялись нафтенаты свинца, но сейчас по экологическим соображениям их почти не используют. По тем же причинам сокращается использование хлорсодержащих присадок. Диалкилдитиофосфаты цинка относятся к наиболее широко применяемым противоизносным присадкам. Диалкилдитиофосфаты цинка – металлоорганические соединения с четырьмя атомами серы, которые координационно связаны с атомом цинка, находящимся в тетраэдрическом гибридизированном состоянии sp^3 . Структура данного химического соединения отражена на рисунке 1 [1].

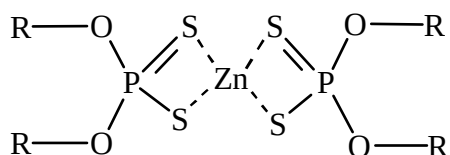


Рисунок 1. – Структура молекулы диалкилдитиофосфата цинка

При выполнении научных исследований разработана технология получения противоизносной присадки с улучшенными трибологическими и антиокислительными свойствами.

По результатам выполненной работы получена присадка RM-22 в результате синтеза на основе регламента производства дитиофосфорной присадки ДФ-11к, но с применением 1,3-диметилбутилового спирта как растворителя.

В результате сравнения рассчитанной по [3] себестоимости присадок ДФ-11к и RM-22 можно заключить, что себестоимость новой присадки значительно выше, а именно на 1826 рублей и 41 копейку, что негативно скажется на стоимости конечного продукта. Но, так как новая присадка обладает лучшими свойствами, то для удовлетворения текущего уровня, который обеспечивает присадка ДФ-11к, необходимо меньшее количество присадки, что приведет к незначительному росту стоимости. Главным преимуществом данной присадки является то, что ее меньшее вовлечение позволяет обеспечить хорошие антиокислительные и противоизносные свойства, а также при этом масло сможет удовлетворять современным жестким требованиям по содержанию элементов в моторных маслах, а именно требованиям групп SM и SN по классификации API на уровне 0,06–0,08 % масс. Р и 0,5–0,7% масс. S, а также требованиям групп C-2, C-3, C-4, C-5 по классификации ACEA. Удовлетворение требованиям данных групп международных классификаций гарантирует высокий спрос на продукцию, так как современные автомо-

били нуждаются в маслах с низким содержанием фосфора и серы для долговечного функционирования систем доочистки отработавших газов, которые предназначены для уменьшения выбросов окислов азота, серы и углеводородов в атмосферу, что способствует решению существующих экологических проблем. Стоит отметить, что данные масла имеют более высокую стоимость и это, в конечном счете, приводит к тому, что высокая себестоимость синтезированной присадки нивелируется за счет ее свойств.

Была рассчитана отпускная стоимость новой присадки. Отпускная стоимость синтезированной присадки ниже, чем у зарубежного аналога Lubrizol LZ-1071. Цена за 1 тонну присадки Lubrizol LZ-1071 составляет 7 280 руб., которая по сравнению с отпускной ценой присадки RM-22, составляющей 7 264,07 руб., выше, что указывает на достаточную экономическую эффективность предлагаемой технологии производства новой присадки.

Важно подчеркнуть, что с экономической точки зрения отпускная стоимость новой присадки RM-22 ниже, чем отпускная стоимость зарубежных аналогов.

Заключение. Обеспечение высоких темпов производства высококачественных смазочных масел за рубежом во многом связано с более масштабным применением высокоэффективных присадок. Производство присадок относится к наукоемким подотраслям нефтеперерабатывающей промышленности, поэтому цены на продукцию присадок превышают в значительной степени показатели по другой малотоннажной продукции (например, базовые масла, смазки) и находятся в прямой зависимости от качества присадок. На основе полученных результатов обоснована эффективность внедрения в производство новой присадки RM-22. Доказана конкурентоспособность полученной присадки на рынке по сравнению с зарубежными аналогами.

ЛИТЕРАТУРА

1. Рудник, Л.Р. Присадки к смазочным материалам. Свойства и применение / Л.Р. Рудник ; пер. с англ. 2-го изд. ; под ред. А.М. Данилова. – СПб. : ЦОП «Профессия», 2013. – 928 с..
2. Кулиев, А.М. Химия и технология присадок к маслам и топливам. – 2-е изд., перераб. / А.М. Кулиев. – Л. : Химия, 1985. – 312 с.
3. Калькуляция себестоимости присадки ДФ-11к ООО «ЛЛК-НАФТАН» за бюджетный период № 42595.

УДК 665.6

ПОЛУЧЕНИЕ БЕЛОГО МАСЛА АДсорбЦИОННОЙ ОЧИСТКОЙ ПРОДУКТОВ ФРАКЦИОНИРОВАНИЯ ОСТАТКА ГИДРОКРЕКИНГА

М.В. СОРОГОВЕЦ, Ю.В. ЛАПИНА
(Представлено: канд. техн. наук А.В. МИТИНОВ)

Описан способ получения белого вазелинового масла адсорбционной очисткой на неподвижном слое адсорбента базового масла гидрокрекинга НС4. Показана возможность получения белого масла с использованием в качестве адсорбента цеолита типа NaX. Исследованы свойства полученного продукта. Изучен процесс регенерации адсорбента прокаливанием.

Современные тенденции развития нефтепереработки связаны, прежде всего, с ее углублением за счет использования новых вторичных процессов.

Основная задача – увеличение выхода нефтяных топлив. Побочные продукты – остаток гидрокрекинга также представляют интерес как потенциальный источник получения новых нефтепродуктов. Специфика процесса гидрокрекинга в том, что гидрогенизационные процессы позволяют увеличить содержание парафинов и нафтенов, что делает целесообразным получения из остатка гидрокрекинга белого масла.

Белые масла – специфическая группа минеральных низковязких масел, получаемых специальной очисткой и применяемых как в технических целях, так и в медицине, фармакологии и косметике. В химической промышленности белые масла применяются в качестве инициаторных – инертного носителя инициатора полимеризации в процессе получения полиэтилена, также используются в качестве редукторного масла и в качестве лампадного.

Традиционным методом получения белых масел является сернокислотная очистка. Метод позволяет получить высокую степень очистки, что позволяет применять продукт даже в медицинских целях, но сопряжен с достаточно высокими затратами на процесс. *Сернокислотная очистка* является самым старым и широко распространенным в нефтяной промышленности методом удаления из масляных дистиллятов асфальто-смолистых веществ, кислород- и серосодержащих соединений и других нежелательных примесей. Этот метод применяется также и для регенерации отработанных масел (обычно высокой степени старения), как моторных с высокоэффективными комплексными присадками, так и сильно окисленных отработанных трансформаторных и других специальных масел.

Для технических целей высокая степень очистки не требуется, что позволяет применить более дешевый процесс. Кроме сернокислотной очистки для отбеливания масел может быть применена гидроочистка, адсорбционная очистка, заключающаяся в извлечении из них смолистых и сернистых соединений, непредельных и ароматических углеводородов путем адсорбции.

В связи с тем, что в Республике Беларусь белые масла не производятся, задача получения технического белого масла на основе отечественного сырья является актуальной.

Основная часть. Белые масла состоят преимущественно из углеводородов парафинового ряда и получают очисткой минеральных нефтяных масел низковязких фракций. Перечень свойств типичных белых масел приведен в таблице 1.

Таблица 1. – Свойства минеральных белых масел

№	Наименование показателей	Масло вазелиновое по ГОСТ 3164	Масло Purity FG WO.	Масло AGIP BIO 12.
1	Вязкость кинематическая при 50 °С	28,0 – 38,5	36,1	69
2	Вязкость кинематическая при 100 °С		5,8	
3	Температура вспышки, определяемая в закрытом тигле, °С, не ниже	185	220	212
4	Плотность при 20 °С, кг/м ³	28,0 – 0,890	–	0,870
5	Индекс вязкости		105	–
6	Зольность, %, не более	0,005	–	–
7	Температура застывания, °С, не выше	–8	–18	–15
8	Содержание воды и парафина	Отсутствие	Отсутствие	Отсутствие
9	Цвет по ASTM, не более		30	30
10	Цвет, условные единицы КНС-1, не более	6		

В качестве сырья для производства белых масел целесообразно выбирать полупродукты, содержащие в основном парафины и нафтенy. Самым близким по уровню свойств базовым маслом, производимым в Республике Беларусь, является масло НС-4 по ТУВУ 300042199.037-2015, которое и было выбрано в качестве объекта исследования. Свойства базового масла НС-4 описаны в таблице 2.

Таблица 2. – Свойства масла НС4 по ТУВУ 300042199.037-2015

№	Наименование показателей	Требования по ТУ ВУ300042199.037-2015	Фактическое значение	Методы испытания
1	Вязкость кинематическая при 100°С	4,00 – 5,00	4,556	СТБ 1798-2007
2	Индекс вязкости, не менее	120	121	СТБ 1797-2007
3	Массовая доля серы, %, не более	0,030	0,015	СТБ 1420-2003
4	Температура текучести, °С, не выше	–12	–14	СТБ 1557-2015
5	Температура вспышки, определяемая в открытом тигле, °С, не ниже	200	230	ГОСТ 4333-87
6	Температура самовоспламенения, °С, не ниже	230	351	ГОСТ 12.1.044-89
7	Массовая доля воды, %, не более	Следы	Следы	ГОСТ 2477-65
8	Потери от испарения, % масс, не более	18,0	10,7	СТБ 1370-2002
9	Цвет по ASTM, не более	1,5	1,0	СТБ 1796-2007
10	Массовая доля механических примесей, %	Отсутствие	Отсутствие	ГОСТ 6370-83
11	Плотность при 15 °С, кг/м ³	823 – 836	824,4	СТБ ИСО 12185-2007
12	Массовая доля селективных растворителей, %, не более	0,3	0,001	ГОСТ 1057-88

Как видно из таблицы 2, по большинству показателей уровень свойств масла НС-4 соответствует уровню свойств белых масел. Единственный параметр, требующий улучшения, это цвет.

Обесцвечивание масла гидрокрекинга проводили адсорбционной очисткой при комнатной температуре на лабораторной установке, включающей закрепленную на штативе делительную воронку с засыпкой слоя адсорбента в приемную колбу. В ходе эксперимента в подготовленную делительную воронку с адсорбентом заливали порцию очищаемого масла. Скорость прохождения масла через адсорбент регулировали краном. Степень очистки оценивали по коэффициенту пропускания при длине волны 490 нм, определяемую на фотоэлектрическом колориметре. В ходе эксперимента исследовались вопросы выбора эффективного адсорбента, определение живучести адсорбента до его насыщения, возможности регенерации адсорбента.

В качестве пробных адсорбентов использовался цеолит NaX, активированный уголь, цеолит А и нефтяной кокс. Опыты проводились при одинаковой скорости потока, ориентируясь на нижнюю границу рекомендаций из литературных источников и высоту слоя 150 мм. Для получения требуемых кондиций по цвету масла очистку проводили в три стадии.

Из рисунка 1 видно, что при выбранной скорости пропускания цвет очищенного масла начинает ухудшаться (коэффициент пропускания снижаться, возвращаясь к исходному значению неочищенного масла). После пропускания объема масла, превышающего шесть объемов загруженного адсорбента, для второй и третьей стадий очистки емкость адсорбента выше.

Полученное после трех ступеней очистки белое масло исследовали стандартными методами. Свойства очищенного масла гидрокрекинга приведены в таблице 3.

Эффективность очистки оценивалась с использованием фотоколориметра по изменению коэффициента пропускания на длине волны 490 нм. Лучшие результаты, приближающиеся к 100 % коэффициенту пропускания, получены только после трехкратной очистки на цеолите Х и активированном угле.

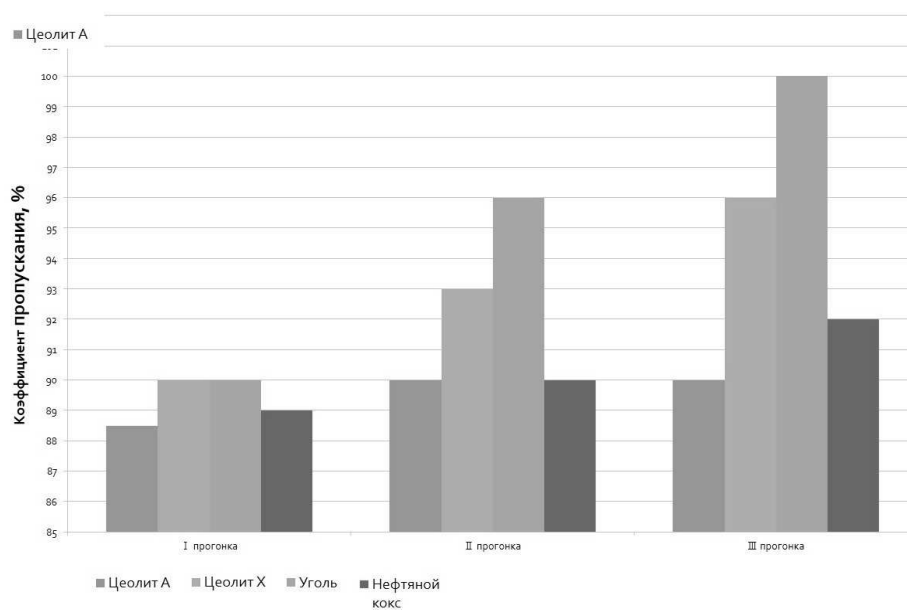


Рисунок 1. – Влияние типа адсорбента на качество адсорбционной очистки

Таблица 3. – Свойства очищенного масла НС-4

№	Наименование показателей	Фактическое значение	Методы испытания
1	Вязкость кинематическая при 40 °С	50,6639	СТБ 1798-2007
2	Индекс вязкости, не менее	120	СТБ 1797-2007
3	Температура застывания, °С	–18	
4	Температура текучести, °С, не выше	–12	СТБ 1557-2015
5	Температура вспышки, определяемая в открытом тигле, °С, не ниже	187	ГОСТ 4333-87
6	Плотность при 15 °С, кг/м ³	830	СТБ ИСО 12185-2007

Емкость адсорбента оценивали, измеряя количество пропущенного через адсорбент масла до момента ухудшения цвета. Результаты эксперимента приведены на рисунке 2.

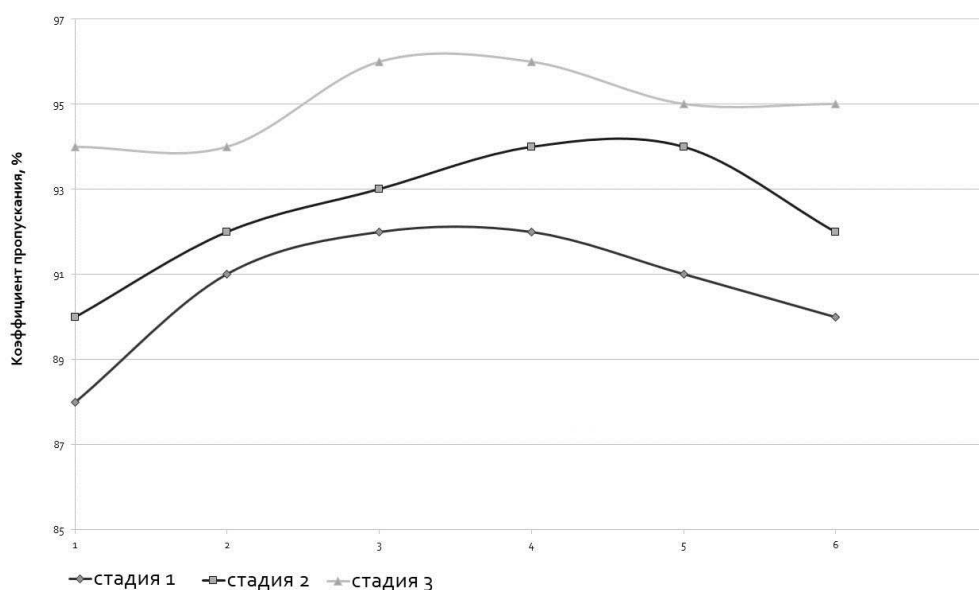


Рисунок 2. – Влияние регенерированного адсорбента на качество очистки

Возможность повторного использования цеолита определяли путем регенерации отработанного адсорбента прокаливанием в муфельной печи. Установлено, что после пяти циклов регенерации адсорбент сохранил свои свойства и обеспечивал получение белого масла с коэффициентом пропускания 99%.

Заключение

Получение белого масла адсорбционной очисткой масла гидрокрекинга является процессом, позволяющим увеличить степень передела углеводородного сырья и внести вклад в повышение эффективности нефтепереработки Республики Беларусь. Полученный продукт соответствует уровню свойств белых технических масел и может быть признан импортозамещающим для ряда направлений использования.

ЛИТЕРАТУРА

1. Суханов, В.П. Переработка нефти / В.П. Суханов. – М. : Высш. школа, 1972.
2. Эрих, В.Н. Химия и технология нефти и газа / В.Н. Эрих, М.Г. Расин, М.Г. Рудин. – М. : Химия, 1977.
3. Альперт, П.З. Основы проектирования химических установок / П.З. Альперт. – М. : Высш. школа, 1989.
4. Шашкин, П.И. Регенерация отработанных нефтяных масел / П.И. Шашкин. – М. : Химия, 1970.
5. Технология переработки нефти. Производство нефтяных масел : учеб.-метод. компл. для студентов спец. 1-48 01 03 «Химическая технология природных энергоносителей и углеродных материалов» и слушателей УО «ПГУ» спец. 1-48 01 72 «Технология переработки нефти и газа» / С.В. Покровская. – Новополоцк : ПГУ, 2008. – 320 с.

УДК 665.6

ОПТИМИЗАЦИЯ РЕЖИМА АДсорбЦИОННОЙ ОЧИСТКИ МАСЛА ГИДРОКРЕКИНГА

М.В. СОРОГОВЕЦ, Ю.В. ЛАПИНА
(Представлено: канд. техн. наук А.В. МИТИНОВ)

Описан подход разработки основных технологических и конструктивных параметров процесса адсорбционной очистки базового масла гидрокрекинга. Для решения задачи выбора оптимальных параметров технологического процесса применен метод планирования эксперимента. Оптимизация процесса адсорбционной очистки масла гидрокрекинга позволит снизить себестоимость получаемого продукта и сделать метод конкурентоспособным.

Адсорбционная очистка масел – это процесс, эффективность протекания которого, зависит от многих параметров. Важнейшими из них являются свойства адсорбента, температура и вязкость среды, в свою очередь, зависящая от температуры и времени контакта среды и адсорбента.

Температура среды определяет интенсивность диффузионных процессов при очистке на молекулярных ситах и является определяющим параметром процесса. Для таких сред, как минеральные масла, температурный фактор важен еще и из-за сильного влияния на вязкость. Для снижения вязкости системы и обеспечения тем самым возможности адсорбционной очистки высоковязких масел применяют прием их разбавления низкокипящими углеводородами. Для приемлемого снижения вязкости низковязких масел достаточно, может быть, повышение температуры. Однако поскольку процесс адсорбции является обратимым, значительное повышение температуры может привести к преобладанию десорбционных явлений и общему снижению эффективности очистки масла. Время контакта среды и адсорбента при достаточной степени очистки определяет экономическую эффективность технологии. При малом времени контакта и высокой скорости пропускания жидкой среды через адсорбент экономическая эффективность может быть перечеркнута неприемлемо низким качеством продукта. Чрезмерно завышенное время контакта обеспечивает гарантированно высокое качество, но может привести к экономической неконкурентоспособности технологии адсорбционной очистки по сравнению с другими технологиями.

Наличие множества параметров процесса адсорбционной очистки масла, некоторые из которых являются взаимозависимыми, и определяет задачу оптимизации режима технологического процесса.

При экспериментальном решении такого рода проблем целесообразно применение метода планирования эксперимента, позволяющего при оптимальном количестве опытов получить математическую модель процесса.

Основная часть. Задача оптимизации технологического процесса адсорбционной очистки масла гидрокрекинга при относительно небольшом количестве управляющих параметров позволяет использовать для реализации экспериментальных исследований дробный факторный эксперимент.

Учитывая имеющийся опыт и рекомендации по температурно-временному режиму адсорбционной очистки масел, выбран диапазон и шаг для исследуемых параметров. Матрица планирования дробного факторного эксперимента по определению влияния на процесс температуры среды и времени контакта приведена в таблице 1.

Таблица 1. – Матрица планирования дробного факторного эксперимента

Температура, t °C	Время, мин					
	10	20	30	40	50	60
20	–	–	–	–	–	–
30	–	–	–	–	–	–
40	–	+	–	+	–	–
50	–	–	+	–	–	–
60	–	+	–	+	–	–
70	–	–	–	–	–	–

Эксперимент по адсорбционной очистке масла приводили с предварительно прокаленным в муфельной печи при температуре 600 °C цеолитом типа NaX марки Z-673TX (2.5) производства бельгийской компании CRI Catalyst Company. В ходе эксперимента порция цеолита заливалась в стеклянную колбу с цеолитом и помещалась в водяную баню. Время контакта определялось с использованием секундомера. Контроль температуры осуществлялся ртутным термометром с ценой деления 0,1°C.

Результаты опытов, проведенных при условиях, определенных матрицей планирования эксперимента, приведены в таблице 2.

Таблица 2. – Значения коэффициента пропускания масла после очистки при режимах дробного факторного эксперимента

Температура, t °C	Время, мин					
	10	20	30	40	50	60
20	–	–	–	–	–	–
30	–	–	–	–	–	–
40	–	0,89	–	0,89	–	–
50	–	–	0,88	–	–	–
60	–	0,90	–	0,91	–	–
70	–	–	–	–	–	–

Обработка результатов эксперимента позволила получить математическую модель процесса.

Для начала исследования была выбрана начальная точка, в которой должны проводиться эксперименты. Обычно эта точка выбирается в середине множества всех точек. Выбрано: $t_0 = 50^\circ\text{C}$, $\tau = 30$ мин.

После кодирования значения факторов получили: $\Delta t = 10^\circ\text{C}$, $x = \frac{t - t_0}{\Delta t}$.

При $t_0 = 50^\circ\text{C}$ для $t = 40^\circ\text{C} = -1$,

$t_{50} = 0$, $t_{60} = +1$; $\Delta\tau = 10$ мин; $\tau = \frac{\tau - \tau_0}{\Delta\tau}$; $\tau_0 = 30$ мин; $\Delta\tau = 10$ мин; $x_2 = \frac{\tau - \tau_0}{10}$; $\tau = 40$ мин $\rightarrow x_2 = 1$;

$\tau = 20$ мин; $x_2 = -1$.

Для обработки результатов построим линейную модель, приведенную в таблице 3.

Таблица 3. – Линейная модель

№ опыта	x_1	x_2	y
1	-1	-1	0,89
2	-1	+1	0,89
3	+1	-1	0,90
4	+1	+1	0,91

$$y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2.$$

Коэффициенты модели рассчитываются по формулам:

$$b_0 = \frac{1}{4} \sum_{i=1}^4 y_i = \frac{1}{4} (0,89 + 0,89 + 0,90 + 0,91) = 0,8995;$$

$$b_1 = \frac{1}{4} \sum_{i=1}^4 y_i \cdot x_{2i} = \frac{1}{4} (-0,89 - 0,89 + 0,90 + 0,91) = 0,0075;$$

$$b_2 = \frac{1}{4} \sum_{i=1}^4 y_i \cdot x_{1i} = \frac{1}{4} (-0,89 + 0,89 - 0,90 + 0,91) = 0,0025.$$

Зависимость прозрачности K от нормализованных факторов:

$$K = 0,8995 + 0,0075 \cdot t_1 + 0,0025 \cdot \tau_2.$$

Линейная зависимость полученной модели предполагает для получения наилучшего результата по целевому показателю смещение обоих параметров к правой границе диапазона. Однако, принимая во внимание физический смысл коэффициента пропускания и асимптотический характер его зависимости от выбранных параметров, целесообразно для оптимизации процесса применить дополнительный, например энергетический или экономический параметр. Такую оптимизацию целесообразно выполнять, используя технологический процесс, приближенный к реальному. Выполнение же исследований в мо-

дельных условиях позволило получить информацию о предпочтительной области основных параметров процесса. И эта область смещена к максимальной границе принятого диапазона.

Проведенные лабораторные исследования позволили установить необходимые температуры и достаточное время контакта. Следует отметить, что на первой стадии оптимизации процесса ни экономические, ни конструкторские факторы не учитывались. На самом деле и поддержание температуры, и обеспечение длительного контакта могут быть связаны с дополнительными материальными затратами на процесс.

Учитывая указанные обстоятельства, выполнен эксперимент по определению конструктивных параметров адсорбера. Модельный лабораторный эксперимент позволил определить оптимальное время контакта в периодическом статическом процессе, причем экономическую целесообразность технология адсорбционной очистки приобретает при организации проточной очистки на неподвижном слое адсорбента. Одно и то же время контактирования единичного объема очищаемого масла может быть реализовано в аппарате малого сечения с большой высотой слоя и высокой (в допускаемых пределах) скоростью потока, либо в аппарате большого сечения с малым слоем и, соответственно, низкой скоростью потока.

Влияние высоты слоя адсорбента на степень очистки определяли на лабораторной установке, включающей закрепленную на штативе делительную воронку с засыпкой слоя адсорбента необходимой высоты и приемную колбу. В ходе эксперимента в подготовленную делительную воронку с адсорбентом NaX марки Z-673TX заливали порцию очищаемого масла. Скорость прохождения масла через адсорбент регулировали краном. Степень очистки оценивали по коэффициенту пропускания при длине волны 490 нм, определяемому на фотоэлектрическом колориметре.

Влияние высоты слоя адсорбента на эффективность очистки масла гидрокрекинга на цеолите NaX приведены в таблице 4.

Таблица 4. – Результаты очистки масла гидрокрекинга при различной высоте слоя адсорбента

Высота слоя адсорбента, мм	Коэффициент пропускания, %
0 (неочищенное масло)	80
150	89
300	90
450	92

Из таблицы 4 видно, что кратное увеличение высоты слоя адсорбента приводит к незначительному повышению чистоты продукта. Однако в ходе эксперимента было отмечено изменение цвета адсорбента по высоте слоя. Наиболее насыщенный адсорбируемыми веществами цеолит темного цвета был в нижней части адсорбера. Верхние слои цеолита имели заметно более светлый цвет.

Таким образом, при большой высоте слоя адсорбента его срабатывание происходит неравномерно, что вызывает необходимость отправлять на регенерацию адсорбент, часть которого еще вполне работоспособна. Учитывая это обстоятельство, представляется целесообразным реализовать процесс адсорбции в несколько последовательно расположенных аппаратах с относительно небольшой высотой слоя цеолита. Применение стадийного подхода для очистки масла позволит рационально использовать возможности емкости цеолита и улучшить экономические показатели процесса.

Заключение

Решение задачи оптимизации технологического режима и конструктивного оформления процесса адсорбционной очистки масла гидрокрекинга позволяет снизить себестоимость получаемого продукта и сделать метод конкурентоспособным по сравнению с традиционно применяемыми. Учитывая относительно небольшую потребность Республики Беларусь в технических белых маслах, метод адсорбционной очистки масла гидрокрекинга, характеризующийся малой капиталоемкостью, может стать предпочтительным.

ЛИТЕРАТУРА

1. Суханов, В.П. Переработка нефти / В.П. Суханов. – М. : Высш. школа, 1972.
2. Эрих, В.Н. Химия и технология нефти и газа / В.Н. Эрих, М.Г. Расин, М.Г. Рудин. – М. : Химия, 1977.
3. Альперт, П.З. Основы проектирования химических установок / П.З. Альперт. – М. : Высш. школа, 1989.
4. Шашкин, П.И. Регенерация отработанных нефтяных масел / П.И. Шашкин. – М. : Химия, 1970.
5. Технология переработки нефти. Производство нефтяных масел : учеб.-метод. компл. для студентов спец. 1-48 01 03 «Химическая технология природных энергоносителей и углеродных материалов» и слушателей УО «ПГУ» спец. 1-48 01 72 «Технология переработки нефти и газа» / С.В. Покровская. – Новополоцк : ПГУ, 2008. – 320 с.

УДК 657.6:66

СРАВНИТЕЛЬНЫЙ ТЕХНИКО-ЭКОНОМИЧЕСКИЙ АНАЛИЗ
ПРОЦЕССОВ ПОЛУЧЕНИЯ СИНТЕЗ-ГАЗА**П.М. ПОЛЯШКЕВИЧ, Е.Я. КОРБУТ****(Представлено: С.П. СТУДЕНИКИНА)**

Рассмотрена актуальность производства синтез-газа. Произведен расчет основных технико-экономических показателей двух принципиально разных установок производства синтез-газа. Показано, что получение синтез-газа путем газификации нефтяного кокса имеет относительно низкую себестоимость и может найти место в схеме нефтеперерабатывающего завода ОАО «Нафтан».

Синтез-газ, представляющий собой смесь водорода H_2 и монооксида углерода CO в различных соотношениях, является сырьем многих химических и нефтехимических продуктов (метанола, аммиака, оксигенатов, синтеза Фишера – Тропша и т.д.), а также используется для восстановления железной руды или в качестве топлива [1].

Первым способом получения синтез-газа была газификация каменного угля – процесс, осуществленный еще в 30-е годы XIX века в Англии. Дальнейшее развитие данной отрасли привело к новым методам получения, среди которых можно выделить конверсию природного газа, газификацию биомасс, твердых бытовых отходов и т.д. [2].

На современном этапе технология производства определяется двумя основными факторами: сырьевой базой (например, в таких странах, как Китай, США, Германия, обладающими крупными угольными бассейнами, преобладает газификация угля; в России же широко используется паровая конверсия природного газа) и дальнейшей переработкой.

В Республике Беларусь получение синтез-газа осуществляется на нефтеперерабатывающих заводах (с последующим концентрированием водорода) из природного газа, ввиду дефицитности или дороговизны других видов сырья. Однако, ввод в эксплуатацию установки замедленного коксования на ОАО «Нафтан» может кардинально изменить сложившуюся ситуацию [3]. Получаемый малоценный продукт – нефтяной кокс, стоимость которого, как правило, в 6–8 раз ниже стоимости перерабатываемой нефти, может быть подвергнут газификации с получением синтез-газа. Таким образом, рассмотрение данного направления становится весьма актуальным.

Целью работы является анализ технико-экономических показателей установки парового риформинга природного газа, действующей на ОАО «Нафтан», и установки газификации нефтяного кокса с последующим рассмотрением экономической целесообразности ее введения.

С технологической точки зрения процессы сильно отличаются между собой: начиная от вида сырья и газифицирующего агента, заканчивая аппаратурным оформлением, поэтому на себестоимость получаемого продукта будут влиять принципиально различные факторы (таблица 1) [4].

Таблица 1. – Сравнение основных технологических показателей процессов

Показатель сравнения	Показатель для процесса:	
	Паровой риформинг	Газификация
Сырье	Природный газ	Нефтяной кокс
Газифицирующий агент	Водяной пар	Паро-воздушная смесь
Молярное соотношение $H_2:CO$ в получаемом газе	(4,8-5,2):1	(0,8-1,1):1
Наличие печи на установке	Имеется	Отсутствие
Отрицательное влияние на экологию	Большее	Меньшее

Расчет технико-экономических показателей проводился согласно общепринятым формулам в экономике предприятия. Результаты расчета представлены в таблице 2.

Таблица 2. – Калькуляция себестоимости на 1000 нм³ синтез-газа для процессов

Показатель сравнения	Значение величины показателя для процесса, белорус. руб	
	Паровой риформинг	Газификация
Сырье и основные материалы	65,30	61,45
Топливо и энергия	0,59	0,25
Оплата труда	0,62	0,55
Отчисление в ФСЗН	0,21	0,19
Показатель сравнения	Значение величины показателя для процесса, белорус. руб	
	Паровой риформинг	Газификация
Общепроизводственные расходы	3,27	2,31
Общехозяйственные расходы	6,51	4,65
Производственная себестоимость	76,50	69,4
Прочие расходы	1,10	0,69
Полная себестоимость	77,60	70,09

На основе полученных результатов расчета можно сделать следующие **выводы**:

- 1) себестоимость продукции во многом определяется стоимостью сырья и основных материалов;
- 2) отсутствие печи на установке газификации способствует уменьшению затрат на топливо;
- 3) технологическая печь является источником организованных выбросов в атмосферу, что приводит к увеличению экологического налога на установке парового риформинга;
- 4) производство синтез-газа методом газификации в 1,11 раза дешевле, чем паровым риформингом.

Таким образом, развитие процессов получения синтез-газа является достаточно перспективным направлением, в том числе и для нашего региона. Внедрение процесса газификации нефтяного кокса позволит получать ценный компонент с более низкой себестоимостью. На нефтеперерабатывающем заводе ОАО «Нафтан» установка по производству синтез-газа может стать заменой или дополнением к имеющимся установкам парового риформинга.

ЛИТЕРАТУРА

1. Ван ден Оостеркамп, П. Достижения в производстве синтез-газа / П. ван ден Оостеркамп, Э. Вагнер, Дж. Росс // Российский химический журнал. – Т. XLIV, № 1. – 2000. – С. 34–42.
2. Технологии получения и переработки синтез-газа / П.Е. Матковский [и др.] // Газохимия. 2011. – № 3–4. – С. 77–84.
3. Открытое акционерное общество «Нафтан» [Электронный ресурс]. – Режим доступа: http://www.naftan.by/ru/default_ru.aspx. – Дата доступа: 15.08.2017.
4. Капустин, В.М. Технология переработки нефти : в 4-х ч. / В.М. Капустин, А.А. Гуреев. – М. : Химия, 2015. – Часть вторая : Физико-химические процессы. – 400 с.

УДК 665.6–403.3

ТЕХНОЛОГИЯ ПРОИЗВОДСТВА, ЭКОНОМИЧЕСКОЕ СРАВНЕНИЕ ИСПОЛЬЗОВАНИЯ БИОГАЗА С ИМЕЮЩИМИСЯ ВИДАМИ ТОПЛИВА ДЛЯ АВТОМОБИЛЬНЫХ ДВИГАТЕЛЕЙ

А.С. КОЛЯГО

(Представлено: С.П. СТУДЕНИКИНА)

Рассматривается технология производства биогаза и сравнительный анализ, показывающий экономическую целесообразность использования биогаза и имеющихся видов топлива для автомобильных двигателей по их характеристикам и экономической рациональности.

Последняя треть XX – начало XXI века характеризуются невиданными темпами роста производственных сил в большинстве стран мира, что привело к резкому увеличению потребления всех видов энергии, в особенности заключённой в ископаемом топливе – угле, нефти и природном газе. В результате этого в ряде стран мира стала ощущаться нехватка традиционных видов топлива, главным образом такого универсального и удобного, как нефть. Энергетический кризис, захвативший многие государства, вызвал огромный рост цены на нефть. Хотя Республика Беларусь на данном этапе и закупает нефть по ценам ниже мировых, но рост также коснулся и нас. Создавшаяся ситуация усилила стремление поставить на службу человеку так называемые нетрадиционные источники энергии, среди которых энергия из биомассы.

Наиболее распространенный способ получения энергии из биомассы – анаэробное (без доступа кислорода) сбраживание отходов сельскохозяйственного производства.

На *первом этапе* анаэробного сбраживания органических веществ путем биохимического расщепления (гидролиза) сначала происходит разложение высокомолекулярных соединений (углеводородов, жиров, белковых веществ) на низкомолекулярные органические соединения.

На *втором этапе* при участии кислотообразующих бактерий происходит дальнейшее разложение с образованием органических кислот и их солей, а также спиртов CO_2 и H_2 , а затем H_2S и NH_3 .

Окончательное бактериальное преобразование органических веществ в CO_2 и CH_4 осуществляется на *третьем этапе* процесса (метановое брожение). Кроме того, из CO_2 и H_2 образуется в дальнейшем дополнительное количество CH_4 и H_2O (рис. 1).

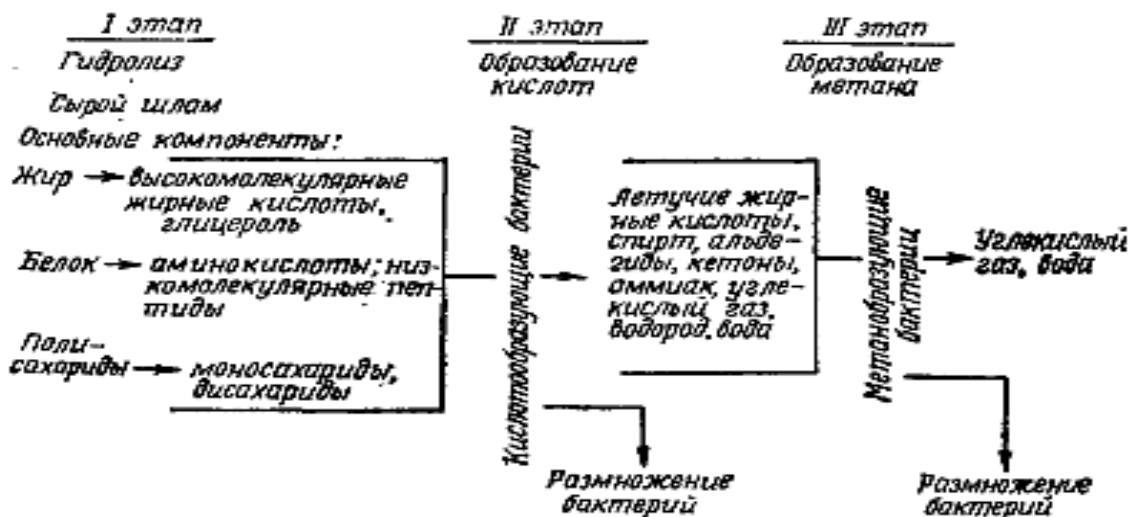


Рисунок 1. – Этапы процесса анаэробного брожения

Получающиеся в результате этого процесса продукты – биогаз и перебродившая полужидкая масса – представляют собой большую ценность как газообразное топливо и органическое удобрение.

К факторам, влияющим на процесс брожения, относят:

- температуру;
- влажность среды;
- уровень pH;
- соотношение C: N: P;

- площадь поверхности частиц сырья;
- частоту подачи субстрата;
- замедляющие вещества;
- стимулирующие добавки.

Так как температура процесса поддерживается от 33 до 54 °С, то не требуется большого расхода энергии на обогрев реактора. Удельный расход энергии на получение 1 м³ биогаза составляет от 3,7 кВт·ч до 5,5 кВт·ч/м³ в зависимости от температуры, необходимой для жизнедеятельности бактерий. От общего производства электроэнергии данной установкой только около 30% идёт на нужды производства (при обогреваемых реакторах). В случае, когда реактора брожения не обогреваются, затраты энергии падают в 1,5 раза [1].

Образующийся в реакторе биогаз скапливается в хранилище и только после дополнительной очистки может быть направлен на получение электроэнергии, тепла, топлива. Содержание метана в полученном биогазе колеблется от 50 до 70% в зависимости от использованного сырья. При подготовке биогаза к использованию учитываются в основном три обстоятельства:

- удаление сероводорода (обессеривание необходимо, прежде всего, для предотвращения коррозии, вызываемой в особенности остаточными продуктами сгорания, и удаления ядовитой части (газовой смеси);
- удаление углекислого газа (повышает теплоту сгорания газа и необходимо для его сжижения);
- компримирование и сжижение (при использовании в качестве топлива).

Есть много способов очистки и обогащения газа. Самым экономичным и увеличивающим извлечение метана из биогаза является разработка Гюнтера Лотара (рис. 2) [2].

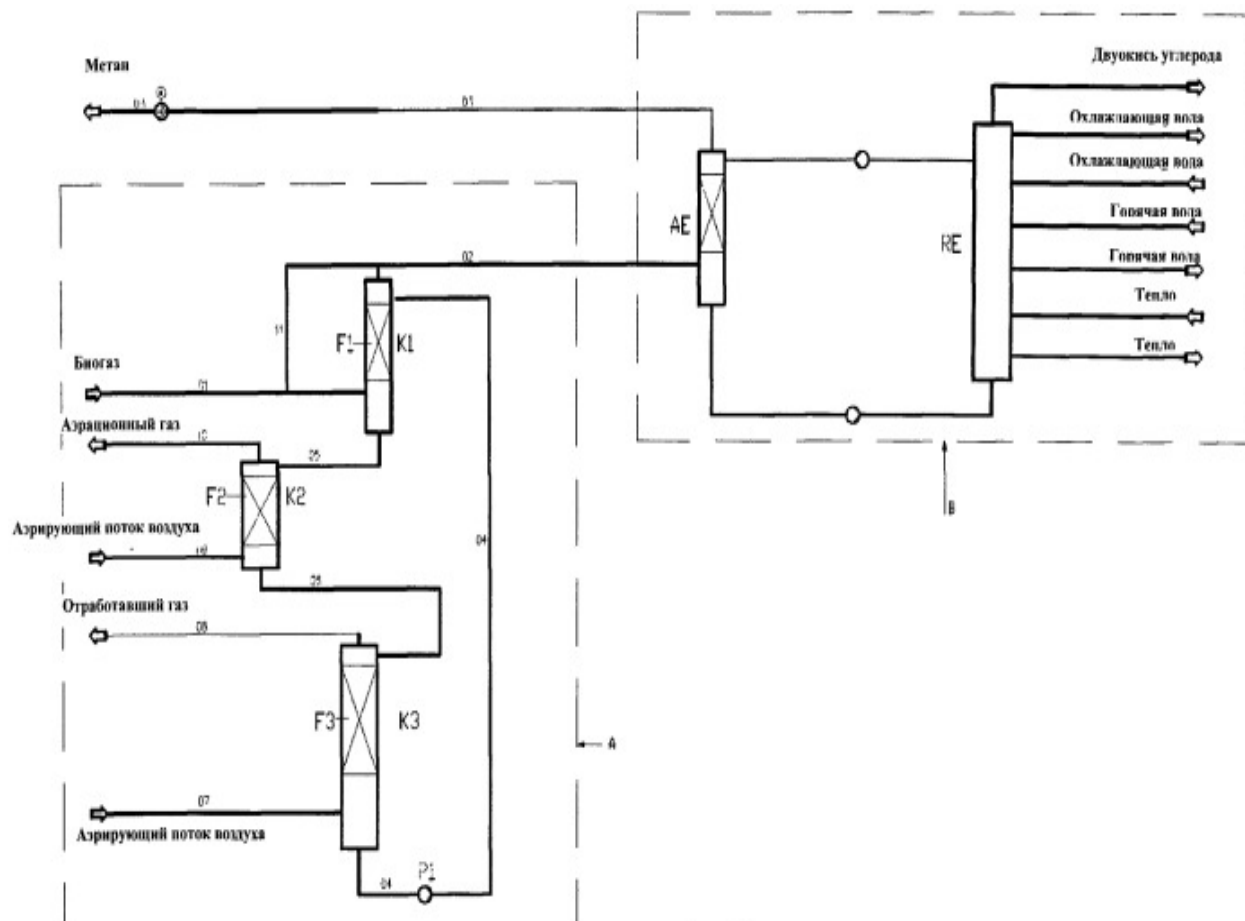


Рисунок 2. – Производство метана анаэробной обработкой биогаза

Смысл очистки заключается в следующем:

- на *первом этапе* биогаз пропускают через очистную колонну (K1) в противоток подаваемой пресной воде, где диоксид углерода, сероводород, аммиак и другие органические водорастворимые вещества связываются в пресной воде, а метановый газ отбирают у головы очистной колонны (K1);

- на *втором* – растворенный метан удаляют в первой испарительной колонне (К2) посредством добавления аэрирующего воздуха или аэрирующего воздуха и кислорода;

- на *третьем* – растворенный диоксид углерода удаляют во второй испарительной колонне (К3) посредством добавления аэрирующего воздуха, при этом отводят очищенный раствор, подаваемый к очистной ступени (К1), и отработанный газ.

Проанализированные методы потребляют относительно большое количество энергии, составляющее 0,5–0,8 кВт·ч/м³ биогаза при нормальных условиях, тем временем как данная система работает при стандартном давлении, потребление энергии для этих трех этапов очистки составляет менее чем 0,03 кВт·ч/м³ при нормальных условиях биогаза, что дает системе возможность работать крайне экономично. Данный способ приводит к сравнительно небольшим потерям метана – до 0,05% при содержании метана в очищенном биогазе до 98%. В результате получаем биометан – полный аналог природному газу, отличие только в происхождении.

Полученный биометан возможно использовать как топливо для автомобилей. По результатам исследований компании РосБиогаз [3] из 1 м³ биогаза можно получить столько же энергии как из 0,48 л бензина, 0,65 л дизельного топлива, 0,6 м³ природного горючего газа. Для сравнения цена за 1 м³ биогаза составляет 0,2 – 0,25\$, за 0,48 л бензина – 0,29\$, за 0,65 л дизельного топлива – 0,41\$.

Биогаз также очень экономичен (табл. 1).

Таблица 1. – Использование биогаза в качестве моторного топлива

Двигатель	Использование	Количество баллонов	Использование биогаза, м ³ /100 км
УАЗ – 469	Автомашина	3	42
ЗИЛ ММЗ-130	Автомашина	9	72

Заключение

Проанализировав технологию получения биогаза, сравнив использование биогаза как автомобильного топлива с уже имеющимся на рынке, можно сделать вывод об экономической целесообразности использования биогаза как автомобильного топлива.

ЛИТЕРАТУРА

1. Баадер, В. Биогаз : Теория и практика : пер. с нем / В. Баадер, Е. Доне, М. Бренндерфер. – М. : Колос, 1982. – 148 с.
2. Способ и система очистки сырых газов, в частности биогаза, для получения метана : пат. 2495706 РФ, МПК-8 B01D53/14, C07C7/11, C02F11/04 [Электронный ресурс] / Гюнтер Лотар (DE) ; заявитель ДГЕ ДР.-ИНЖ. ГЮНТЕР ИНЖИНИРИНГ ГМБХ (DE) ; заявл. 22.05.2009 ; опубл. 20.10.2013. – Режим доступа : <http://www.freepatent.ru/patents/2495706>.
3. Биогаз – новый путь в будущее! [Электронный ресурс] / Биогаз и биогазовые установки. – Пермь, Екатеринбург, Москва, 2010-2012. – Режим доступа: <http://www.rosbiogas.ru>. – Дата доступа: 01.09.2017.

УДК 620.952

ЭКОНОМИЧЕСКОЕ СРАВНЕНИЕ УСТАНОВОК ПО ПРОИЗВОДСТВУ БИОГАЗА С АЛЬТЕРНАТИВНЫМИ ИСТОЧНИКАМИ ДЛЯ ВЫРАБОТКИ ЭЛЕКТРОЭНЕРГИИ

А.С. КОЛЯГО

(Представлено: С.П. СТУДЕНИКИНА)

Выполнен сравнительный анализ установок по производству биогаза с другими альтернативными источниками энергии и предложен выбор наиболее экономичного и целесообразного варианта для получения электроэнергии.

Растущая мировая экономика ежегодно увеличивает потребление энергии на 2–3%. Источником производства последней до сих пор остаются нефть, природный газ, уголь. Республика Беларусь не является страной, экономика которой обеспечена собственными энергетическими ресурсами. Соответственно она относится к числу энергозависимых от внешних поставок энергоносителей.

По данным статистики [1] цены на нефть с каждым годом будут интенсивно увеличиваться. Цены на газ для Республики Беларусь падали до 2015 года, в 2016 году – 132 долл. за 1000 м³, 2017 году – 141,11 долл. за 1000 м³ [2]. Проблемы энергетической зависимости Республики Беларусь можно решать разными путями, одним из которых является использование возобновляемых источников энергии. В ходе исследования были проанализированы следующие источники энергии, которые могут подходить для выработки электрической энергии в Республике Беларусь:

- энергия ветра;
- солнечная энергия;
- энергия геотермальных источников;
- турбодетандерные установки;
- гидроэнергетика;
- биогаз.

Рассмотрим их подробнее со всеми достоинствами и недостатками.

На территории Республики Беларусь выявлено 1840 площадок для размещения ветроустановок с теоретически возможным энергетическим потенциалом более 1600 МВт. Имеющиеся до недавнего времени разработки, позволяющие преобразовывать энергию ветра в электроэнергию, в условиях Республики Беларусь малоэффективны. Однако современные технические решения позволяют создавать ветроэнергетические установки с пусковой скоростью ветра от 3 м/с и номинальной скоростью эксплуатации 7–8 м/с. Но цена на такие установки для промышленности в зависимости от мощности варьируется от 20 000 € до 170 000.

Использование солнечной энергии целесообразно в странах, где наблюдается большое количество ясных дней. По метеорологическим данным в Республике Беларусь в среднем 150 дней в году пасмурных, 185 – с переменной облачностью и 30 ясных дней, а среднегодовое поступление солнечной энергии на земную поверхность с учетом ночей и облачности составляет 243 кал на 1 см² в сутки, что эквивалентно 2,8 кВт·ч на м² в сутки. К основным направлениям использования энергии солнца можно отнести сельскохозяйственное производства и бытовые цели.

Опыт использования геотермальных ресурсов в Беларуси уже имеется. Извлекаемые ресурсы геотермальной энергии имеются в зоне распространения пресных вод на всей территории Республики Беларусь. Однако их плотность изменяется в широком диапазоне от приблизительно 10 – 12 кг у.т./м² (интервал глубины 100–200 м) до 300–350 кг у.т./м² в глубоко погруженных участках кембрийских отложений в Брестской области. Преимуществом использования геотермальной энергии из зон, насыщенных пресными водами, является то, что после снятия тепла эти воды не требуют их возврата в подземные горизонты и могут быть использованы для различных практических нужд, в том числе и для питьевого водоснабжения. Это удешевляет сооружение геотермальных установок и снижает сроки их окупаемости [3].

Биогаз – газ, получаемый водородным или метановым брожением биомассы. Перечень органических отходов, пригодных для производства биогаза: навоз, птичий помет, послеспиртовая барда, пивная дробина, свекольный жом, фекальные осадки, отходы рыбного и забойного цеха, трава, бытовые отходы, отходы молокозаводов, отходы производства биодизеля – технический глицерин от производства биодизеля из рапса, отходы от производства соков, водоросли и др. Также биогаз можно производить из специально выращенных энергетических культур, например, из силосной кукурузы. Выход газа может достигать до 300 м³ из 1 тонны.

Полученный биогаз можно использовать как сам по себе (в качестве автомобильного топлива), так и для получения электроэнергии, пара в паровых котлах, тепла в водогрейных котлах, которые работают на биогазе.

Преимущества использования биогазовых установок:

- рентабельное использование материала вместо затратного уничтожения отходов;
- дополнительный доход для производства компоста и ферментов;
- быстрая окупаемость расходов;
- низкие затраты на эксплуатацию;
- низкая стоимость обслуживания и износа техники;
- получение попутных продуктов – удобрений;
- улучшение экологической обстановки вокруг животноводческих комплексов.

Так как Республика Беларусь является по большому счету сельскохозяйственной страной, то наиболее выгодное замещение импортируемым энергоносителям это биогаз. На 2017 год планируется поставить из России в Беларусь 2,5–2,8 млрд кВт·ч электрической энергии. Это 23,1% от потребляемой электроэнергии в год. Количество вырабатываемых отходов сельского хозяйства в год – 48 млн тонн. Исходя из расчета того, что из 1 тонны отходов возможно получить до 300 м³ газа (для расчета было принято 50 м³), то в год можно получить 4,008 млрд кВт·ч электрической энергии. В результате видна перспектива полного замещения поставок электроэнергии из других стран. В Республике Беларусь в 2007 году была запущена пилотная установка общей мощностью 1,1 МВт. Для переработки всех сельскохозяйственных отходов, получаемых в год, нужно около 3644 таких установок, при общей площади в 500 м². Установки могут занять приблизительно 2 км². Для сравнения – площадь города Новополоцка 48,49 км².

Так же был произведен расчет объема капитальных затрат на строительство установки по производству биогаза. За основу была взята установка Био-350 объемом перерабатываемых отходов 350 т/сутки, производством энергии 90000 кВт·ч/сутки. В результате получен объем капитальных затрат в размере 3,7 млн \$. Цена на электроэнергию из биогаза будет ниже за счет относительно низкой себестоимости продукции.

Также из плюсов биогаза можно отметить простоту технологии производства.

Среди плюсов производства биогаза есть и минусы:

- высвобождение парниковых газов, связанное с использованием азотных удобрений;
- загрязнение фосфатными удобрениями земель и грунтовых вод;
- орошение;
- вытеснение производства продовольственных культур;
- возможный рост цен на продукты питания.

Заключение

В Республике Беларусь прослеживается тенденция к полному отказу от импортной электрической энергии, что может напрямую сказаться на эффективности и положительной динамике развития экономики нашей страны. Сравнив экономические показатели для установок по производству биогаза с другими альтернативными источниками энергии, можно сделать вывод о целесообразности использования биогаза для выработки электроэнергии.

ЛИТЕРАТУРА

1. Всемирный банк [Электронный ресурс] / Группа Всемирного банка. – 2017. – Режим доступа: <http://www.vseмирnyjbank.org>. – Дата доступа: 07.05.2017.
2. Маненок, Т. Почему Беларуси сегодня сложнее торговаться с «Газпромом»? / Т. Маненок // Белрынок [Электронный ресурс]. – 2017. – Режим доступа: <http://www.belrynok.by/2016/01/15/pochemu-belarusi-segodnya-slozhnee-torgovatsya-s-gazpromom/>. – Дата доступа: 08.05.2017.
3. Смольская, Н.А. Возобновляемая энергетика в Беларуси: оценка состояния и перспективы роста / Н.А. Смольская // Науч. тр. Белорус. гос. экон. ун-та : юбил. сб. Вып. 6 / М-во образования Респ. Беларусь, Белорусский гос. экон. ун-т ; [редкол.: В.Н. Шимов (пред.) [и др.]. – Минск : БГЭУ, 2013. – С. 363–369.

УДК 665.6/7

ОСНОВНЫЕ ТЕХНОЛОГИИ ПОЛУЧЕНИЯ СИНТЕЗ-ГАЗА И ЕГО МЕСТО В СОВРЕМЕННОМ МИРЕ

П.М. ПОЛЯШКЕВИЧ, Е.Я. КОРБУТ
(Представлено: С.П. СТУДЕНИКИНА)

Рассмотрена актуальность производства синтез-газа в современном мире. Описаны основные процессы его получения, достоинства и недостатки. Показано, что широкое разнообразие процессов позволяет получать синтез-газ даже в тех странах, которые имеют ограниченный доступ к различным природным ресурсам.

Обеспечение и рациональное применение энергоресурсов и нефтехимического сырья являются обязательными условиями успешного развития экономики любой современной страны. Основными источниками природного органического сырья служат нефть, природный газ, уголь, горючие сланцы и т.д. В настоящее время наиболее востребованными энергохимическими ресурсами являются нефть и природный газ, добыча которых приближается к максимально возможной [1].

На территории стран СНГ данные виды ресурсов используются в основном только для производства нефтепродуктов или непосредственно в качестве топлива, не находя широкого распространения в нефтехимии. Поэтому доля нефтехимической отрасли в российском ВВП составляет около 1,5%, в то время как в Германии и Китае этот показатель находится на уровне 7 и 8% соответственно. По прогнозам специалистов, доля продукции нефтехимии на мировом рынке будет неуклонно увеличиваться [2].

Ситуация, аналогичная России, наблюдается и в Беларуси. Однако стоит отметить, что в последние годы на белорусском нефтехимическом рынке наметились позитивные сдвиги: государство проявляет внимание к происходящим в отрасли процессам. Так, например, в государственной программе развития Республики Беларусь на 2016–2020 годы [3] указано, что ориентирами в химической и нефтехимической промышленности должны быть не только материало- и энергосберегающие технологии, но и производства широкого спектра синтетических и композиционных материалов, в том числе новых поколений.

В сложившейся для Беларуси ситуации популярность стали набирать процессы, позволяющие расширить сырьевую базу нефтехимии. Среди такого рода процессов все большую актуальность приобретают процессы производства синтез-газа, переработка которого позволяет получить широкий набор продукции (таблица 1) [4].

Таблица 1. – Удельная потребность в синтез-газе для производства основных нефтехимических продуктов

Продукт	Удельная потребность в синтез-газе, $\text{нм}^3/\text{т}$
Метанол	0,300
Уксусная кислота	0,065
Уксусный ангидрид	0,039
Продукты оксосинтеза	0,104
Муравьиная кислота	0,078
Метилформиат	0,067
Пропионовая кислота	0,053
Метилметакрилат	0,104

Здесь стоит отметить, что мировой спрос только на метанол в 2015 году составлял порядка 45 млн тонн в год, а к 2021 году прогнозируется увеличение до более 80 млн тонн [5].

Синтез-газ представляет собой смесь монооксида углерода CO и водорода H_2 в различных соотношениях. Он является одним из главных источников сырья для промышленного органического синтеза.

На современном этапе существует несколько основных технологий получения синтез-газа, которые требуют применения высоких температур, давлений и большинство из них, использования катализаторов. Все эти процессы можно классифицировать по виду газифицируемого сырья или по виду газифицирующего агента.

Газификация угля. Наибольшее распространение получил метод газификации угля парокислородным дутьем. Процесс основан на реакциях:



Реакция (1) является эндотермической, поэтому равновесие сдвигается вправо при температурах 900–1000 °С. Реакция сгорания угля (2) является экзотермической и обеспечивает нужный тепловой баланс. Процесс проводится в специальном аппарате – газификаторе.

Основные достоинства процесса производства синтез-газа:

- процесс не требует применения катализаторов;
- возможность переработки широкого диапазона углеродистого сырья, причем в последнее время по указанной технологии производится также газификация нефтяного кокса [6];
- процесс является экологически чистым, так как для его проведения нет необходимости в технологических печах;

- низкая стоимость технологического оборудования, основное из которых газификатор и скруббер.

Основные недостатки:

- как правило, для проведения процесса помимо дешевого водяного пара необходимо применять дорогой чистый кислород, чтобы облегчить очистку синтез-газа;
- получаемый синтез-газ обеднен водородом.

Паровая конверсия метана. В основе процесса лежит реакция:



Условия процесса: температура в пределах 800–1000 °С, давление порядка 2–2,5 МПа. Процесс каталитический – проводится на никелевом катализаторе в трубчатой печи с внешним обогревом.

Основные достоинства:

- метод довольно давно используется в промышленности, как следствие большой опыт эксплуатации и широкий выбор поставщиков базового оборудования;
- высокое содержание водорода в получаемом синтез-газе.

Основные недостатки:

- использование дорого чувствительного катализатора определяет необходимость подготовки сырья;
- процесс протекает в трубчатой печи, следовательно необходимы существенные затраты метана на обогрев реакторных трубок открытым пламенем для поддержания требуемой температуры.

Парциальное окисление метана. Процесс основан на реакции:



Процесс протекает при температурах 1200–1500 °С и давлении 3–4 МПа в среде недостатка кислорода в турбинных генераторах с предокислительной камерой.

Основные достоинства:

- возможность обеспечения собственной энергией, как вывод – отсутствие технологической печи;
- различные виды парциального окисления могут проводиться как в присутствии катализатора, так и без него;
- для процесса характерны наименьшие эксплуатационные затраты [7];
- низкие удельные нормы расхода сырья и энергии [8].

Основные недостатки:

- необходимость высоких капитальных вложений;
- необходимость использовать чистый кислород, затраты на получение которого составляют до 40 % от стоимости синтез-газа [9].

Автотермический риформинг метана. Процесс представляет собой экзотермическое парциальное окисление с эндотермической паровой конверсией метана при температурах 1300–1600 °С и давлении до 15 МПа по суммарной реакции:



Основные достоинства:

- процесс является некаталитическим;
- низкие эксплуатационные затраты.

Основные недостатки:

- процесс является относительно новым и обладает малой промышленной распространенностью;
- требуется использование дорого оборудования ввиду жестких условий процесса;
- для протекания процесса необходимо использование чистого кислорода;
- невысокий процент выхода продукта.

Углекислотная конверсия метана. Синтез-газ можно также получать взаимодействием CO_2 с природным газом. Такой метод часто называют «сухой» конверсией:



Основные достоинства:

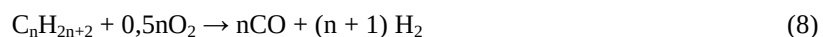
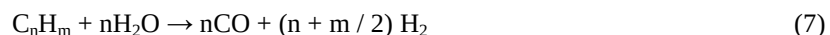
- в процесс вовлекается углекислый газ, запасы которого огромны, а масштабы использования в промышленности невелики;

- относительно высокий выход синтез-газа.

Основные недостатки:

- требует большого расхода топлива на эндотермическую реакцию;
- использование дорогих катализаторов, склонных к закоксуыванию.

Газификация биомассы и твердых бытовых отходов. Данный метод основывается на том, что большое количество отходов являются углеводородными материалами или их производными [1]. Поэтому под действием высоких температур в среде газифицирующего агента можно получить синтез-газ:



Состав получаемого продукта зависит от условий (температура в пределах 1350–1450 °С, давление от 2 до 14 МПа) и типа сырья.

Основные достоинства:

- сырьем выступают различного рода отходы, значит, переработка их в синтез-газ является крайне экологичным вариантом;

- процесс протекает без присутствия катализаторов;
- газифицирующий агент подбирается по доступности.

Основные недостатки:

- необходимость значительных инвестиционных затрат;
- невысокий выход синтез-газа и крайне затруднителен процесс его очистки;

Рост потребления химической и нефтехимической продукции все больше требует развития процессов, позволяющих расширить сырьевую базу. В условиях изменяющегося рынка получение синтез-газа является довольно перспективным решением, так как он является основой для получения широко ассортимента продукции, а разнообразие методов его получения позволяет выбрать наиболее эффективный для данных условий.

ЛИТЕРАТУРА

1. Технологии получения и переработки синтез-газа / П.Е. Матковский [и др.] // Газохимия. – 2011. – № 3–4. – С. 77–84.
2. Нефтехимия в России: выбор вектора развития [Электронный ресурс]. – Режим доступа: [http://www.ey.com/Publication/vwLUAssets/EY-petrochemical-industry-in-russia-2015-rus/\\$FILE/EY-petrochemical-industry-in-russia-2015-rus.pdf](http://www.ey.com/Publication/vwLUAssets/EY-petrochemical-industry-in-russia-2015-rus/$FILE/EY-petrochemical-industry-in-russia-2015-rus.pdf). – Дата доступа 17.08.2017.
3. Программа социально-экономического развития Республики Беларусь на 2016–2020 годы [Электронный ресурс]. – Режим доступа: http://www.government.by/upload/docs/program_ek2016-2020.pdf. – Дата доступа: 17.08.2017.
4. Ван ден Оостеркамп, П. Достижения в производстве синтез-газа / П. ван ден Оостеркамп, Э. Вагнер, Дж. Росс // Российский химический журнал. – 2000. – Т. XLIV, № 1. – С. 34–42.
5. Метанол 2016 – итоги отраслевой конференции [Электронный ресурс]. – Режим доступа: <https://mplast.by/novosti/2016-07-06-metanol-2016-itogi-otraslevoy-konferentsii/>. – Дата доступа: 18.08.2017.
6. Капустин, В.М. Технология переработки нефти : в 4-х ч. / В.М. Капустин, А.А. Гуреев. – М. : Химия, 2015. – Ч. вторая. Физико-химические процессы. – 400 с.
7. Седнин, В.А. Повышение энергетического потенциала биогаза / В.А. Седнин, О.Ф. Краецкая, И.Н. Прокопья // Изв. высш. учеб. заведений и энергетических объединений СНГ – Энергетика : междунар. науч.-техн. журн. – 2012. – № 1. – С. 73–77.
8. Галанов, С.И. Парциальное каталитическое окисление природного газа в синтез – газ / С.И. Галанов, К.А. Косырева, Е.А. Литвак // Вестн. Томского гос. ун-та. – 2012. – № 364. – С. 230–233.
9. Окислительная переработка низших алканов: состояние и перспективы / Н.Я. Усачев [и др.] // Российский хим. журн. – 2008 – Т. LII, № 4. – С. 22–31.

УДК 662.758.2

ОПРЕДЕЛЕНИЕ ФРАКЦИОННОГО СОСТАВА БИОДИЗЕЛЬНЫХ ТОПЛИВ И ИХ СМЕСЕЙ**Е.А. ТУРОВЕЦ***(Представлено: канд. техн. наук, доц. А.В. СПИРИДОНОВ)*

Рассматриваются биодизельные добавки (растительные масла) к моторному топливу. Проведено исследование эффективности применения и свойств биодизельных топливных композиций. Представлен их фракционный состав. Приведены основные характеристики, проанализирован мировой опыт использования отходов растительных масел.

Идея использовать растительные масла в качестве топлив для дизельных двигателей была выдвинута еще при их создании (первый двигатель с воспламенением от сжатия был построен Рудольфом Дизелем в 1897 году). Однако с освоением нефтяных запасов в XX веке более выгодным оказалось топливо из нефти. Сейчас биодизельное топливо часто отождествляют с рапсовым маслом, которое действительно стало основным сырьевым источником «биодизеля» в Европе, но биодизельное топливо можно получать и из других масел, например, подсолнечного, пальмового или соевого.

Важно иметь в виду, что сами по себе растительные масла в качестве топлив не используются. Любое биодизельное топливо представляет собой смесь растительных масел. В растительном масле содержатся жиры – эфиры жирных кислот с глицерином. В Европе основным биодизельным топливом стал метиловый эфир рапсового масла.

Растительные масла и их эфиры, как и спирты, отличаются агрессивностью ко многим материалам, традиционно используемым в двигателях и топливной системе автомобилей. В последние годы большинство европейских производителей выпускают машины, допускающие использование смесей нефтяного топлива с биодизельным в количестве 5–20%, а иногда и 100% биотоплива. Добавление биодизельного компонента в количестве до 5% обычно считается приемлемым для любых двигателей, неадаптированных к биотопливу. Достаточно активно биодизельное топливо внедряется и в США, где в качестве сырья используют чаще всего соевое масло. Еще один перспективный источник биодизельного топлива – отработанные пищевые масла.

Истощение нефтяных месторождений и продолжающийся рост цен на нефть и нефтепродукты обуславливают неизбежное и все более широкое использование в дизельных двигателях биотоплив на основе растительных масел. Перевод дизелей на биотоплива позволит не только обеспечить замещение топлив нефтяного происхождения топливами, производимыми из возобновляемых сырьевых ресурсов, но и заметно снизить токсичность отработавших газов и улучшить экологическую ситуацию в городах и сельской местности [1–3].

Для централизованного снабжения топливом автотранспорта в значительной степени пригодны сложные эфиры растительных масел – метиловые, этиловые и бутиловые, получаемые путем обработки растительных масел спиртами (соответственно метанолом, этанолом и бутанолом).

Из перечисленных эфиров менее доростоящими являются метиловые эфиры. Причем сложные эфиры растительных масел можно получать из различных растительных масел – рапсового, подсолнечного, соевого, пальмового и других. Возможно использование для этих же целей животных жиров.

Производство биодизельных топлив на основе растительных масел и животных жиров постоянно расширяется. Так, в 2005 году выпуск биодизельного топлива в странах ЕС составил около 3,5 млн т, в 2009 году его объем достиг 12 млн т, а в 2011 году – около 18 млн т этого топлива. Данные о динамике производства биодизельного топлива в некоторых странах ЕС представлены на рисунке 1 [1].

Биодизельные топлива, предназначенные для использования в дизельных двигателях, должны за минимальный период сформировать в камере сгорания топливно-воздушную смесь, обеспечивающую ее легкое воспламенение, плавное и достаточно полное сгорание с минимальным содержанием сажи, токсичных и канцерогенных веществ. Не допускается образование осадков в топливоподающей системе и нагароотложений на деталях двигателя.

Исходя из изложенного, одной из основных проблем является организация рабочего процесса дизельных двигателей, работающих на биодизельных топливах. Обеспечение современных требований к показателям топливной экономичности и токсичности возможно лишь при дальнейшем совершенствовании процессов распыливания топлива и смесеобразования [2–5]. При этом скорость формирования топливно-воздушной смеси предопределяется как длиной и шириной топливных струй, так и их структурой (качеством распыления), интенсивностью испарения топлива и диффузии его в сжатый воздух.

Для достижения требуемого качества процесса смесеобразования топливо должно обладать заданным *фракционным составом*, который является одним из важнейших показателей эксплуатационных

свойств топлив для двигателей внутреннего сгорания. Фракционный состав характеризует содержание в топливе различных фракций, выкипающих в определенных температурных пределах.

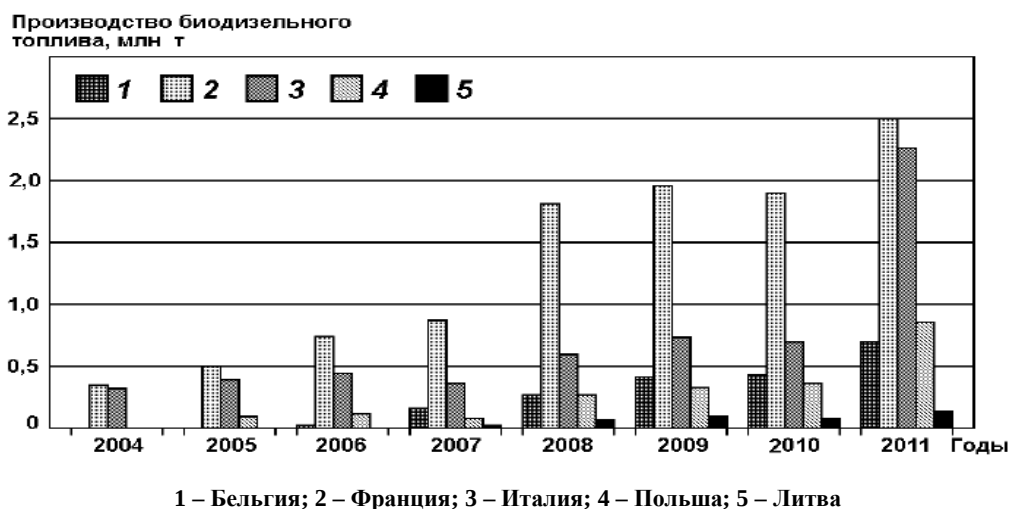


Рисунок 1. – Динамика роста производства биодизельного топлива в странах Европы

При снятии характеристик фракционного состава для традиционного дизельного топлива обычно выделяют температуру начала перегонки (начала кипения), температуры перегонки 10, 50, 90% топлива и температуру окончания перегонки (конца кипения), соответствующую перегонке 96 или 98% топлива. Температура перегонки 10% топлива характеризует склонность топлива к образованию паровых пробок в системе питания дизеля. Наличие в топливе большого количества легких фракций приводит к увеличению жесткости сгорания (скорости нарастания давления и максимального давления сгорания) и снижению ресурса работы дизельного двигателя.

Среднюю испаряемость топлива устанавливают по температуре выкипания 50%, которая обуславливает испаряемость топлива и период задержки воспламенения [4, с. 13–94]. Наличие в топливе тяжелых трудноиспаряющихся фракций можно определить по температуре 90%-ной точке кипения при перегонке. По температуре окончания перегонки, соответствующей перегонке 96 или 98% топлива, также выявляют присутствие в нем тяжелых (выкипающих при высоких температурах) фракций, чрезмерное содержание которых в топливе приводит к ухудшению испарения и усложнению образования однородной топливно-воздушной смеси. Высокая температура выкипания 96% топлива свидетельствует о наличии тяжелых фракций, которые не только ухудшают смесеобразование, но и увеличивают нагарообразование на деталях цилиндропоршневой группы, снижая тем самым экономичность и надежность работы двигателя. При использовании топлива с большим содержанием тяжелых фракций некоторая их часть догорает в такте расширения, вызывая дымность отработавших газов и повышение их температуры, рост удельного расхода топлива, загрязнение моторного масла и деталей двигателя продуктами неполного сгорания топлива, прежде всего сажей.

Перечисленные выше факторы указывают на необходимость учитывать фракционный состав при выборе топлива для дизелей. Это также относится и к биодизельному топливу. Фракционный состав топлив определялся методом перегонки – путем разделения смеси жидкостей на компоненты, основанного на разности температур их кипения. Данный метод заключается в нагревании жидкости до кипения с последующей конденсацией паров в холодильнике.

В настоящее время существуют способы и устройства автоматического определения фракционного состава топлив. На рисунке 2 изображен автоматический аппарат AD86 5G для определения фракционного состава светлых нефтепродуктов [6].

На рисунке 3 приведены кривые фракционной разгонки нефтяного ДТ и различных растительных масел [1; 7; 8]. Температура кипения ДТ находится в пределах 140 до 350 °С. Температура начала кипения биодизельных топлив, полученных из различных растительных масел, обычно колеблется от 220 до 285 °С. При количестве испарившегося топлива более 10% кривые разгонки сложных метиловых эфиров идут резко вверх. При повышении температуры выше $t = 310...330$ °С метиловые эфиры жирных кислот подвергаются термическому разложению.



Рисунок 2. – Автоматический анализатор фракционного состава светлых нефтепродуктов AD86 5G (Франция)

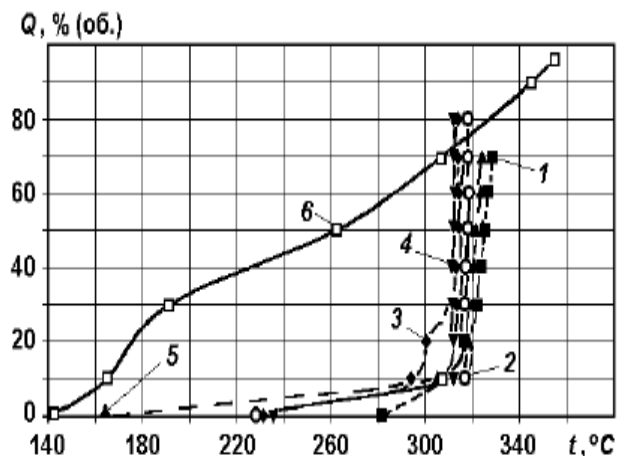


Рисунок 3. – Фракционный состав дизельного топлива (6) и биодизельных топлив, полученных из масел: рапсового (1), кукурузного (2), подсолнечного (3), льняного (4), редьки (5)

При перегонке биодизельного топлива отмечено термическое разложение его компонентов. Можно предположить, что оно происходит и при работе дизельного двигателя. Образующиеся продукты окисления и полимеризации являются причиной негативных процессов в двигателе – нагаро- и коксоотложения на деталях камеры сгорания. Температура выкипания 50% фракций для нефтяного ДТ в данном случае составляла 265 °C. Для всех биодизельных топлив эта температура намного выше – находится в интервале 300...330 °C.

С этой точки зрения целесообразно использовать в качестве моторного топлива смеси нефтяного ДТ и метиловых эфиров растительных масел. В смесях с нефтяными ДТ указанные негативные свойства метиловых эфиров растительных масел в значительной степени нивелируются: эти компоненты хорошо смешиваются друг с другом в любых пропорциях с образованием стабильных смесей.

Заключение. Приведенные данные экспериментальных исследований подтверждают возможность использования в дизелях биодизельных топлив, получаемых из различных сырьевых ресурсов. Наибольшее приближение к свойствам нефтяных дизельных топлив обеспечивает применение смесей нефтяного дизельного топлива и метиловых эфиров растительных масел. При этом удастся получать показатели токсичности отработавших газов, которые заметно лучше аналогичных показателей дизеля, работающего на чистом дизельном топливе. Причем улучшения показателей токсичности отработавших газов достигают даже при небольшом содержании биодизельного топлива в смеси (5–10%). Это позволяет использовать исследованные биодизельные топлива в качестве экологических добавок к нефтяному дизельному топливу.

ЛИТЕРАТУРА

1. Использование биотоплив на основе растительных масел в дизельных двигателях / В.А. Марков [и др.] // Вестн. МГТУ им. Н.Э. Баумана. Сер. «Машиностроение». – М., 2012.
2. Использование растительных масел и топлив на их основе в дизельных двигателях / В.А. Марков [и др.] М. : ООО НИЦ «Инженер», 2011. – 536 с.
3. Зачем нужны биотоплива [Электронный ресурс]. – Режим доступа: <http://computerra.ru/> - Компьютерра-Онлайн.htm. – Дата доступа: 13.04.2015.
4. Топлива, смазочные материалы, технические жидкости. Ассортимент и применение : справочник / под ред. В.М. Школьников. – М. : Техноинфор, 1999.
5. [Электронный ресурс]. – Режим доступа: // [http://car188.ru.](http://car188.ru/) – Режим доступа: 27.08.2014.
6. Спиридонов, А.В. Определение фракционного состава бензинов с кислородсодержащими высокооктановыми компонентами / А.В. Спиридонов, Д.П. Панфилов // Вестн. Полоц. гос. ун-та. Серия, С. Фундаментальные науки. – 2010. – № 9. – С. 142–149.
7. ASTM D86. Standard Test Method for Distillation of Petroleum Products at Atmospheric Pressure, 1999, 22 с.
8. Нефтепродукты. Методы определения фракционного состава : ГОСТ 2177-99. – 25 с.

УДК 66-97

**ПИРОМЕТРИЧЕСКИЕ МЕТОДЫ КОНТРОЛЯ КИПЕНИЯ
В СОВРЕМЕННЫХ АВТОМАТИЧЕСКИХ АНАЛИЗАТОРАХ
ФРАКЦИОННОГО СОСТАВА СВЕТЛЫХ НЕФТЕПРОДУКТОВ**

Е.А. ТУРОВЕЦ*(Представлено: канд. техн. наук, доц. А.В. СПИРИДОНОВ)*

Рассматривается возможность вычисления температур паров выкипающего нефтепродукта в колбе стандартизованного аппарата простой постепенной перегонки пирометрическим методом, что позволит скорректировать нагрев испытуемых нефтепродуктов в автоматическом режиме и обеспечить более четкое выполнение требований стандартных методик без необходимости ручной корректировки нагрева оператором.

Фракционный состав нефтепродуктов определяют в топливных лабораториях НПЗ. Сущность метода (по ГОСТ 2177-99 и ASTM D86) [1; 2] заключается в перегонке 100 мл образца с последующей регистрацией показаний термометра и объёма отогнанного нефтепродукта в приёмном мерном цилиндре. В настоящее время этот метод широко используется в ряде лабораторий НПЗ в странах СНГ.

Более прогрессивным методом является автоматический метод определения фракционного состава нефтепродуктов (Automatic). В настоящее время на международном рынке используются автоматические анализаторы, производимые американской корпорацией Petroleum Analyser Corporation (PAC), которая стремится быть монополистом на рынке приборов. На рисунке 1 представлен прибор Optidist, производимый этой корпорацией.

Анализ фракционного состава любых светлых нефтепродуктов ведется в полностью автоматическом режиме без участия лаборанта, который лишь наполняет специальный мерный цилиндр, переливает в колбу, устанавливает стеклянное оборудование в прибор. После программной проверки соответствия установленного оборудования выбранному продукту и правильности монтажа всех узлов запускается полностью автоматический цикл анализа, включающий контроль температуры нагревателя, времени до начала кипения, температуры первой капли (начало кипения), скорости разгонки, температуры паров, конца кипения по падению температуры паров или датчику последней капли.

Прибор не требует никаких предварительных тестов или настроек, оптимальные параметры предварительного нагрева и собственно анализ ведется с первой попытки в полностью автоматическом режиме в соответствии с выбранным стандартом.

Автоматическая оптимизация процесса нагрева позволяет не только улучшить эксперимент, но и защищает материал колбы от излишнего перегрева. Это положительно сказывается на сроке эксплуатации дистилляционных колб и нагревателя, уменьшает риск возникновения пожара вследствие разрушения колбы во время анализа. В современных автоматических анализаторах фракционного состава светлых нефтепродуктов возникает необходимость пирометрического контроля температур нагрева исходных проб испытуемых веществ, что особенно важно в случае загрузки в колбу анализатора продукта с малоизвестными свойствами. Это позволит избежать перегрева и «бурного вскипания» легкокипящих, а также ускорить нагрев высококипящих нефтепродуктов и обеспечить выполнение требований стандартных методик перегонки по времени падения первой капли (начало кипения) в автоматическом режиме без необходимости ручной регулировки нагрева оператором.

Для разработки методики пирометрического управления нагревом предварительно проведём эксперимент с эталонным индивидуальным углеводородом на приборе Optidist. Результаты эксперимента представлены на рисунках 2–3.



Рисунок 1. – Автоматический анализатор фракционного состава нефтепродуктов «Optidist»

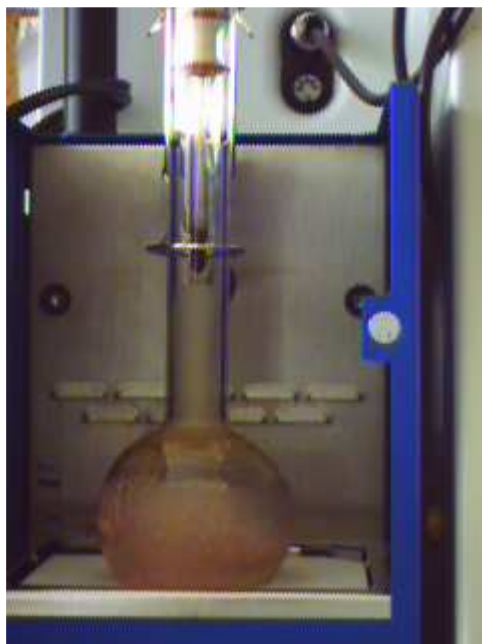


Рисунок 2. – Процесс кипения углеводородов в колбе прибора Optidist (фотоснимок)

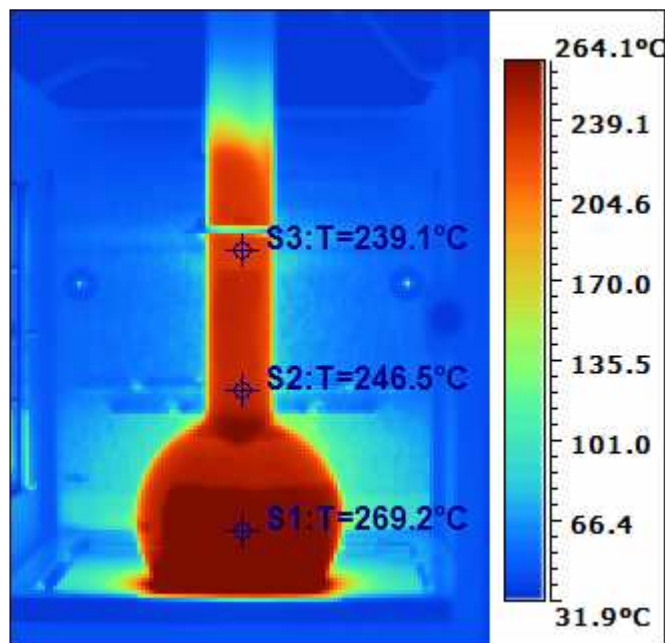


Рисунок 3. – Фотоснимок, полученный с помощью тепловизора (пирометра)

На рисунке 4 представлено распределение температур по времени проведения анализа пробы эталонного вещества (гексадекана). В интервале 0–650 с (> 10 мин) происходит нагрев пробы, в интервале 650–2270 с (38 мин) наблюдается изотермический участок. Это свидетельствует о наступлении динамического равновесия между конденсирующимся паром и окружающей средой (воздухом).

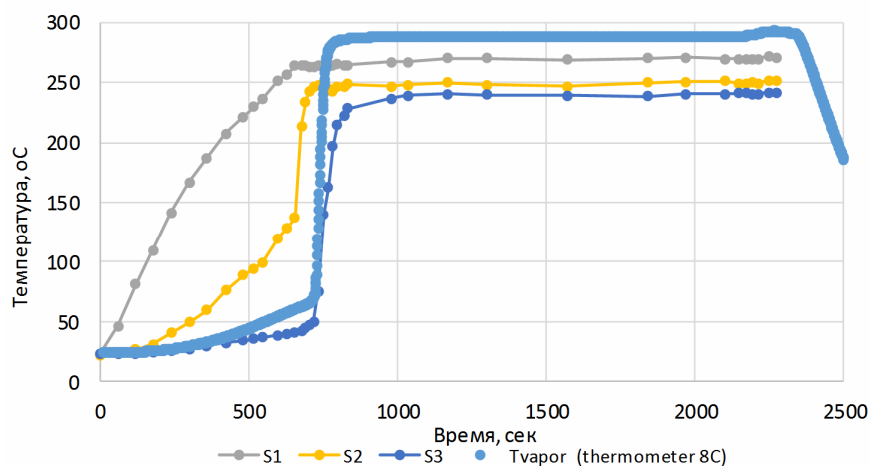


Рисунок 4. – Распределение температур по времени проведения анализа пробы эталонного вещества (гексадекана)

Для разработки математического описания тепловых процессов в колбе составим основное уравнение теплопередачи [3; 4]:

$$Q = KF\Delta t, \quad (1)$$

где F – поверхность теплопередачи, м^2 ; Δt – средняя разность температур между конденсирующимся паром и окружающим воздухом, K – коэффициент теплопередачи.

Плотность теплового потока (теплонапряженность)

$$q = \frac{Q}{F} = K\Delta t. \quad (2)$$

На основе обработки экспериментальных данных получены численные значения коэффициентов теплоотдачи конденсации углеводородных паров на стенке горловины колбы в точках S2 и S3. Они примерно составляют

$$K_{S1} = 102 \text{ Вт/(м}^2\cdot\text{К)}, K_{S2} = 128 \text{ Вт/(м}^2\cdot\text{К)}, K_{S3} = 330 \text{ Вт/(м}^2\cdot\text{К)}.$$

Подставив значения коэффициентов теплопередачи, предварительно преобразовав формулу (2) можно вычислить вероятную температуру испытуемого продукта в конкретных точках колбы.

$$tv_s = \frac{q}{K_s} + t_s, \quad (3)$$

где t_s – температура по показаниям пирометра.

На основе формулы 3 построим зависимости температур продукта в колбе в различных точках от показаний пирометра (рис. 5).

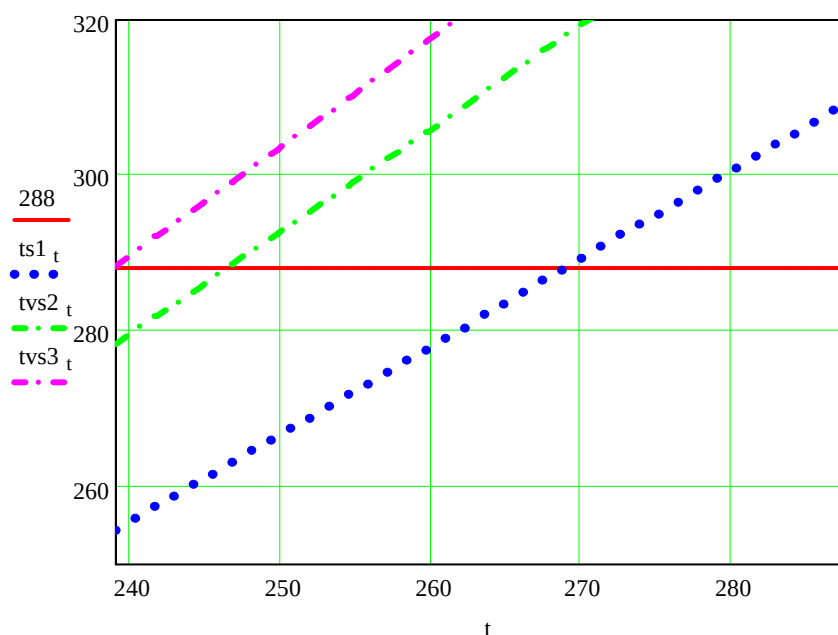


Рисунок 5. – Зависимости вероятных температур нефтяных фракций от показаний пирометра в контрольных точках колбы

По оси абсцисс графика – значения пирометра, по оси ординат – значения температур продукта в колбе. Прямые отображают зависимости этих температур в соответствующих контрольных точках (S1, S2 и S3). Из графика видно, что пунктирные прямые пересекают красную контрольную линию, соответствующую температуре паров в колбе (288 °С) в точках, близких к своим экспериментальным значениям.

Вывод. Так как полученные расчетные значения температур соответствуют экспериментальным значениям температуры 288 °С, то по результатам расчётов можно судить о том, что данную математическую модель можно использовать для расчета температур испытуемого нефтепродукта по значениям пирометра. Таким образом, согласно графику (см. рис. 5), по показаниям пирометра можно определить вероятную температуру паров в колбе (распределение температур). На основании данной методики можно скорректировать программу управления нагревом проб испытуемого вещества в автоматических анализаторах фракционного состава светлых нефтепродуктов.

ЛИТЕРАТУРА

1. ASTM D86 : Standard Test Method for Distillation of Petroleum Products at Atmospheric Pressure, 1999. – 22 с.
2. Нефтепродукты. Методы определения фракционного состава : ГОСТ 2177-99. – 25 с.
3. Исаченко, В.П. Теплопередача : учебник для вузов / В.П. Исаченко, В.А. Осипова, А.С. Сукомел. – 4-е изд. пер. и доп. – М. : Энергоатомиздат, 1981. – 416 с.
4. Справочник химика / под ред. Б.П. Никольского. – М.–Л. : Химия, 1968. – Т. V. – 963 с.

УДК 628.543

ОБЗОР И СРАВНЕНИЕ РЕАКТОРОВ-ЭЛЕКТРОКОАГУЛЯТОРОВ

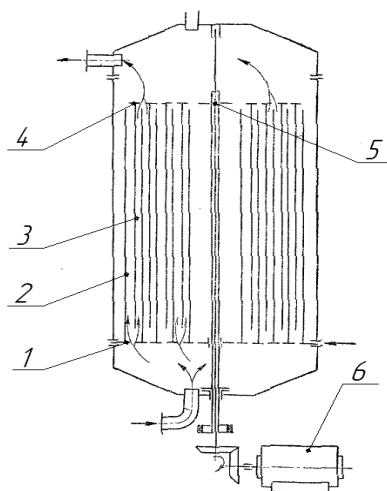
А.М. ЖУРИДОВ

(Представлено: Л.П. ШЕВЧЕНКО)

Проводится краткое сравнение реакторов-электрокоагуляторов различной конструкции. Показаны их достоинства и недостатки. На основании проведенного анализа предлагается аппарат, устройство и принцип работы которого описываются.

Очистка загрязненных вод методом электрокоагуляции основана на электролизе с использованием металлических (железных или алюминиевых) анодов, подвергающихся электрохимическому растворению. Вследствие растворения металлов вода обогащается соответствующими ионами, образующими затем в нейтральной или слабощелочной среде гидроксид соответствующего металла, который и является собственно коагулянтom. Для проведения процесса коагуляции, с использованием электрического тока, применяют реакторы, называемые электрокоагуляторами. Существует большое их разнообразие, каждый из которых имеет свои достоинства и недостатки. При проектировании таких аппаратов стремятся уменьшить расход электроэнергии, повысить эффективность и максимально автоматизировать процесс, что ведет к усложнению конструкции, а вместе с тем и к увеличению стоимости аппарата. Как результат, некоторые из предложенных вариантов не нашли своего материального воплощения, но представляют интерес для изучения и создания более совершенных технических решений.

Типовой электрокоагулятор включает в себя емкость (корпус аппарата) с размещенными в ней анодным и катодным блоками, имеющими однотипную конструкцию и состоящие из металлических пластин [1]. Из *преимуществ* следует отметить простоту аппарата, легкость монтажа электродных блоков. К *недостаткам* следует отнести сравнительно высокую стоимость электродов-пластин, возможность их коробления. В результате протекания процесса образуются гидроокиси металлов, которые вызывают пассивацию электродов. В этом случае возникнет дополнительное сопротивление между электродами, и для дальнейшего проведения процесса нужно будет увеличивать напряжение в цепи. Наиболее простым способом депассивации электродов является *поочередная смена их полярности*. Однако в этом случае электролитическому растворению будут подвергаться оба электрода (катод и анод), что может наложить некоторые ограничения на конструкцию аппарата. Второй метод заключается в *создании турбулентного движения жидкости на границе соприкосновения с электродом*. Это достигается тем, что в предлагаемый электрокоагулятор (рис. 1) установлен опорный диск с неподвижно закрепленными на нем электродами отрицательной полярности, имеющий наклонные отверстия, направленные по винтовой линии, а цилиндрические электроды положительной полярности сверху укреплены на вращающемся колесе с центральным приводным валом. Вращение осуществляется от электродвигателя [2].

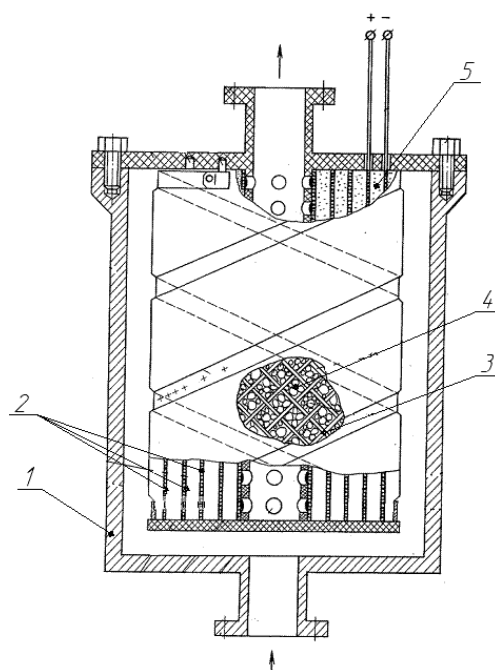


1 – опорный диск; 2 – электроды (анодные); 3 – электроды (катодные);
4 – вращающееся колесо; 5 – приводной вал; 6 – электродвигатель

Рисунок 1. – Электрокоагулятор с вращающимися электродами

Устранение пассивации электродов позволит повысить рабочую плотность тока в 2–3 раза, а вместе с тем и производительность электрокоагулятора. Но не стоит забывать и о недостатках: во избежание коробления электродов необходимо будет проводить замену анодного блока раньше, чем через 50% от его полного износа, необходима высокая точность центрации пар электродов для сохранения постоянства зазора между ними и предотвращения их соприкосновения. К тому же появляется двигатель и уплотняющий элемент, что приведет к дополнительным проблемам при обслуживании.

В целях ускорения электролиза, повышения качества обработки жидкости предлагается использовать **электрокоагулятор с фильтрующими элементами**, электроды которого выполнены из электролитически нерастворимого материала в виде сетки. Ее ячейки заполнены полидисперсными частицами аноднорастворимого металла, а между электродами помещены упругие фильтрующие слои, преимущественно из поролона (рис. 2). Аппарат состоит из корпуса, разнозаряженных проницаемых электродов, выполненных из сетки с полидисперсными частицами. Между электродами находится фильтрующий элемент [3].



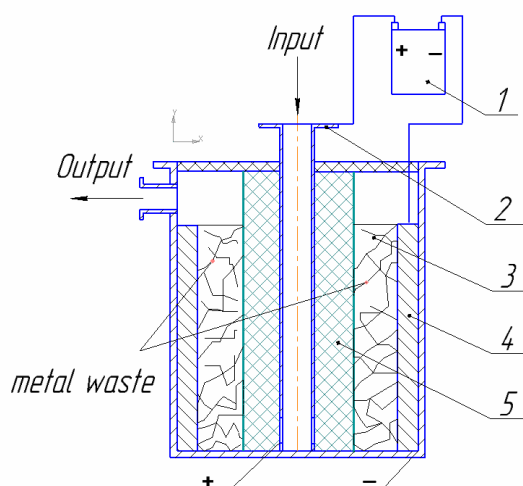
1 – корпус; 2 – разнозаряженные проницаемые электроды;
3 – сетка; 4 – полидисперсные частицы; 5 – фильтрующий элемент

Рисунок 2. – Электрокоагулятор с фильтрующими элементами

Недостатком данного электрокоагулятора являются дорогостоящие комбинированные электроды, высокие гидравлические сопротивления для жидкости. При данном способе необходимо будет часто прибегать к регенерации фильтрующих элементов путем противотока либо к замене всего электродного блока.

Устройство предлагаемого нами электрокоагулятора заключается в использовании дешевых насыпных электродов, которые можно будет формировать из собственных металлических отходов предприятия, взамен дорогостоящих электродов-пластин. Эскиз предлагаемого электрокоагулятора типа реактор с насыпным электродом представлен на рисунке 3.

Аппарат состоит из регулируемого источника тока, к которому подключены электроды: катод в виде трубы и насыпной анод. Анод представляет собой конструкцию из металлического кольца, внутренняя поверхность которого контактирует с металлическими отходами. Для предотвращения соприкосновения металлических отходов с катодом и установления необходимого зазора между электродами используется разделительная сетка, выполненная из диэлектрика. Зазор между анодом и катодом устанавливается аналогично случаю с металлическими электродами-пластинами. Анодное кольцо максимально удалено от зоны интенсивного растворения, которая находится у границы разделительной сетки, и практически не разрушается. Корпус аппарата желательно выполнять из материалов, не проводящих электрический ток (бетон, полимеры), либо изолировать электроды от соприкосновения со стенками.



1 – источника тока; 2 – катод в виде трубы; 3 – насыпной анод;
4 – металлическое кольцо; 5 – разделительная сетка

Рисунок 3. – Электрокоагулятор с насыпным электродом

Очищаемая жидкость через трубообразный катод поступает в нижнюю часть электрокоагулятора. Поднимаясь вверх через насыпной анод, она завихряется и движется в турбулентном режиме по всему объему аппарата из-за хаотичного расположения металлических объектов. Это позволяет устранять пассивацию катода. После некоторого промежутка времени жидкость отводится в блок отстаивания через верхний штуцер.

Заключение

Описанный электрокоагулятор недорогостоящий и простой в обслуживании и эксплуатации, позволяет ускорить электролиз благодаря более развитой удельной поверхности электродов.

Актуальность использования предлагаемого нами аппарата состоит в следующем – позволяет решить проблему закупки дорогостоящих электродов-пластин и предоставляет возможность вовлечения металлических отходов в производственный процесс, что скажется на уменьшении себестоимости выпускаемой продукции.

ЛИТЕРАТУРА

1. Стахов, Е.А. Очистка нефтесодержащих сточных вод / Е.А. Стахов. – Л. : Недра, 1983.
2. Электрокоагулятор непрерывного действия : пат. 407844 / Э.А. Кизлер ; заявитель Урал. науч.-исслед. проектный институт медной промышленности ; заявл. 01.11.1971 ; опубл. 01.01.1973 / ФИПС. База патентов СССР.
3. Электрокоагулятор : пат. № 537956 / О.В. Смирнов, Ю.С. Веселов, И.С. Лавров ; заявитель предприятие А-1097 Ленингр. инженерно-строительный ин-т ; заявл. 10.02.1975 ; опубл. 05.12.1976 / ФИПС. База патентов СССР.

СОДЕРЖАНИЕ

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

<i>Дунченко А.И.</i> Семантические фреймы: классификаторы и квалификаторы	3-7
<i>Дунченко А.И.</i> Семантические фреймы: разработка формата сериализации данных.....	8-11
<i>Родченков А.Н.</i> Особенности проектирования базы данных веб-приложения «Автоматизированная информационная система ведения отчетности для ОАО «НАФТАН»».....	12-14
<i>Родченков А.Н.</i> Проектирование функциональной части веб-приложения «Автоматизированная информационная система ведения отчетности для ОАО «Нафтан»»	15-17
<i>Андрейчиков А.Г.</i> Комбинированные алгоритмы шифрования, целесообразность их использования для защиты файлов от несанкционированного доступа	18-20
<i>Андрейчиков А.Г.</i> Проектирование программного продукта для защиты от несанкционированного доступа к файлам	21-22
<i>Вечеров А.В.</i> Проектирование графического интерфейса для мобильного приложения «Музыканты России» под операционную систему iOS.....	23-25
<i>Вечеров А.В.</i> Проектирование базы данных для мобильного приложения «Музыканты России» под операционную систему iOS.....	26-28
<i>Губко А.В.</i> Разработка системы управления интернет-магазином, взаимодействие системы с пользователем.....	29-31
<i>Губко А.В.</i> Экономическая составляющая разработки интернет-магазина, оценка влияния информационной среды на экономику	32-34
<i>Иванов И.И.</i> Обнаружение лиц на изображениях методом гистограммы направленных градиентов	35-36
<i>Иванов И.И.</i> Классификация данных методом опорных векторов	37-38
<i>Карабанов Р.Ю.</i> Симметричные и асимметричные шифры.....	39-41
<i>Карабанов Р.Ю.</i> Проблематика популярных алгоритмов шифрования	42-43
<i>Купцов В.С.</i> Обзор систем управления базами данных, основанных на MySQL	44-45
<i>Купцов В.С.</i> ERP-система как инструмент для структурирования и автоматизации процесса управления предприятием	46-48
<i>Лаптев Ю.В.</i> Проблемы параллелизма в программировании.....	49-50
<i>Лаптев Ю.В.</i> Принципы и приемы обработки очередей.....	51-52
<i>Левицкий А.А.</i> Проектирование REALTIME веб-приложений при помощи WEBSOCKET и серверных событий.....	53-54
<i>Левицкий А.А.</i> Разработка комплексной системы управления проектами ИТ-компаний	55-57
<i>Лобанов А.В.</i> Программное обеспечение, применяемое для сбора конфиденциальной информации средствами электронной почты	58-59
<i>Лобанов А.В.</i> Методы защиты от средств слежения за электронной почтой.....	60-61
<i>Медюхо Д.А.</i> Мобильная база данных REALM как альтернатива SQLITE и COREDATA для операционной системы IOS.....	62-63
<i>Медюхо Д.А.</i> Шифрование пользовательских данных в базе данных REALM для операционной системы IOS.....	64-65
<i>Меницкий Е.А.</i> Подходы к разработке интерфейса для веб-приложения по работе с медицинскими амбулаторными данными пациентов	66-68
<i>Меницкий Е.А.</i> Проектирование веб-приложения для работы с медицинскими амбулаторными данными пациента	69-71
<i>Соловьёв А.А.</i> Библиотека глубинного обучения DEEPLARNING 4J.....	72-74
<i>Соловьёв А.А.</i> Обнаружение дыма от пожара с применением обработки на основе свёрточных нейронных сетей	75-77
<i>Шевцов М.А.</i> Способы машинного обучения	78-79
<i>Шевцов М.А.</i> Задачи, решаемые с помощью машинного обучения	80-81
<i>Юрцевич Н.В.</i> Анализ функциональных возможностей мобильных приложений, предназначенных для изучения английского языка	82-83
<i>Юрцевич Н.В.</i> Разработка мобильного приложения для изучения английского языка «HELLO-IT-WORLD»	84-86
<i>Дрозд Д.В.</i> Использование Apache Cordova как платформы для разработки мобильных приложений с открытым исходным кодом в связке с Фреймворком Ionic.....	87-88
<i>Дрозд Д.В.</i> Разработка мобильного приложения «JoinUs», предназначенного для организации развлекательных и культурно-общественных мероприятий	89-90

Дысин А.В. Анализ функциональности электронных-путеводителей на примере мобильного приложения и WEB-сайта	91-94
Дысин А.В. Особенности реализации интерактивного 3D-путеводителя по городу Полоцку.....	95-97
Рачицкий О.И. Виды аутентификации.....	98-99
Рачицкий О.И. Уязвимости в современном клиент-серверном общении.....	100-101
Малашиенок С.Н. Целесообразность использования технологии ASP.NET и её платформы WEBAPI при разработке ВЕБ-приложений.....	102-103
Малашиенок С.Н. Разработка интерфейса WEB-приложения для оптимизации формирования графиков отпусков сотрудников компании	104-105
Кохановский А.В. Проектирование стегосистемы, основанной на сокрытии текстовых данных в аудиофайлах за счет изменения времени задержки эхо-сигнала	106-107
Кохановский А.В. Определение степени пригодности контейнера для стеганографической модификации в стегосистеме, основанной на сокрытии текстовых данных в аудиофайлах за счет изменения времени задержки эхо-сигнала	108-109
Карцевич Р.А. LibGDX – кроссплатформенный фреймворк	110-111
Карцевич Р.А. Создание 2D-анимации объектов.....	112-113
Апанчѐнок К.А. Программное обеспечение контроля посещаемости студентов на основе анализа отпечатков пальцев.....	114-116
Апанчѐнок К.А. Особенности аппаратно-программного взаимодействия системы контроля посещаемости студентов на основе анализа отпечатков пальцев	117-120
Дементьев В.Ю. Особенности проектирования базы данных системы управления афишей культурных событий Республики Беларусь	121-122
Дементьев В.Ю. Проектирование клиентской части системы управления афишей культурных событий Республики Беларусь	123-125
Дѐкин Д.Ю. Использование Liquibase для управления изменениями схемы базы данных	126-127
Дѐкин Д.Ю. MyBatis как инструмент для работы с базами данных в Java-приложениях	128-131
Статут А.И. Сравнительный анализ использования облачных решений и собственных серверов при разработке веб приложений	132-133
Статут А.И. Сравнительный анализ методов и средств разработки мобильных приложений	134-136
Шеринѐв Н.О. Web-scraping как инструмент извлечения полезных данных из сети Интернет	137-138
Шеринѐв Н.О. Мобильное приложение «Музыканты России» для операционной системы Андроид ...	139-140
Голован К.С. Понятие технологии блокчейн	141-144
Голован К.С. Прогрессивные веб-приложения.....	145-148
Войтехович С.С. Анализ сложности вопроса в системе для проведения адаптивного тестирования	149-150
Войтехович С.С. Использование электронных систем обучения для контроля знаний	151-153
Пирович А.И. Фреймворк SemanticUI	154-155
Пирович А.И. Технология Flexbox.....	156-157
Гусевский Р.П. Моделирование данных в NOSQL СУБД на примере разработанного веб-приложения «Информационный агрегатор банков Республики Беларусь»	158-160
Гусевский Р.П. Выбор технологий и разработка адаптивного веб-приложения «Информационный агрегатор банков Республики Беларусь»	161-163
Григорян К.Х. Оптимизация архитектуры баз данных с целью повышения надёжности и скорости работы	164-166
Григорян К.Х. Автоматическое получение сертификатов для организации протоколов HTTPS	167-168
Мисевич И.В. Построение веб-приложений с помощью трехслойной архитектуры	169-170
Мисевич И.В. Сравнение фабрик. Фабричный метод, абстрактная фабрика и другие.....	171-173
Иванов М.А. Нейронные сети и мобильные приложения.....	174-175
Иванов М.А. Технологии работы нейронных сетей.....	176-178
Станкевич П.А. Разработка уровня работы с базой данных с использованием Фреймворка MYBATISB веб-приложении «Онлайн сервис для обучения программирования на JAVA».....	179-184
Станкевич П.А. Внедрение системы авторизации и аутентификации пользователей на основе фреймворка SPRINGSECURITYB веб-приложении «Онлайн сервиса для обучения программирования на JAVA»	185-187
Кувецкий Ю.С. Защита информации, передаваемой по протоколу SOAP. Безопасность WEBсервисов (WSSECURITY).....	188-189
Кувецкий Ю.С. Особенности реализации WEB-сервисов, отвечающих за удаленную компиляцию и выполнение программного кода.....	190-191
Валькович А.И. Целесообразность использования игровых сервисов, предоставляемых APPLE для разработки приложений.....	192-193

Валькович А.И. Удаленная настройка параметров приложения. Сравнение собственного решения и FIREBASE REMOTE CONFIG	194-196
Емельянов Г.В. Обзор средств создания системы дополненной реальности	197-199
Емельянов Г.В. Принципы проектирования графических объектов виртуальной реальности	200-202
Ваишкевич М.Ю. Разработка нейросети для управления беспилотным аппаратом	203-204
Ваишкевич М.Ю. Нейронные сети: что они могут сейчас и что смогут в будущем.....	205-206
Дивин К.П. Проектирование мобильного приложения – органайзера по уходу за комнатными растениями.....	207-208
Дивин К.П. Разработка интерфейса мобильного приложения органайзера, по уходу за комнатными растениями.....	209-210
Балабаиш М.А. Использование Socket.io для асинхронного и симметричного обмена данными.....	211-213
Балабаиш М.А. WebAssembly – путь к новым горизонтам производительности в Web.....	214-216
Резанов С.В. Использование cloudkit в качестве средства синхронизации данных	217-219
Резанов С.В. Локализация приложения на операционной системе IOS	220-222
Гайдель А.П. Выбор PHP фреймворка	223-224
Гайдель А.П. Способы оптимизации баз данных при работе с MYSQL	225-226
Змитрович С.Ю. Устройство управления теплицей на основе микроконтроллера	227-229
Змитрович С.Ю. Микроконтроллеры во встроенных системах управления	230-232
Сметана Е.В. Виды рассеяния сигналов в оптических кабелях	233-235
Сметана Е.В. Измерение характеристик оптических кабелей с помощью рефлектометров	236-237
Левощенко В.Н. Взаимодействие радиоимпульсных сигналов со средой над углеводородными залежами.....	238-240
Левощенко В.Н. Моделирование взаимодействия импульсных сигналов со средой над углеводородами.....	241-244
Криштопа В.А. Конструкция испытательного устройства для определения упругих свойств материалов	245-247
Криштопа В.А. Датчик линейного перемещения резистивного типа	248-250
Пастиков С.В. Модернизация станка рядовой намотки ряд-01.....	251-253
Становой А.О. Проектирование GSM антенны	254-256
Волков А.Н., Кошелева Ю.В. Анализ эффективности использования методов Рунге – Кутты разного порядка для решения задач строительной механики	257-260
Кошелева Ю.В., Волков А.Н. Применение метода Рунге – Кутты 4-го порядка для решения задач строительной механики	261-263
Игнатьева С.А. Фокусировка радиолокационных изображений с использованием пакета matlab ..	264-267
Родченкова Е.А. Расчет освещенности в программном пакете DIALUX	268-269
Родченкова Е.А. Светотехнический расчет освещения цеха в программном пакете DIALUX	270-273
Чупров Н.Д. Система экстренного оповещения для людей с ограниченными возможностями	274-276
Борисенко В.Ю. Разработка лабораторного стенда по изучению основ автоматизации студентов специальностей 43 01 03 «электроснабжение» и 39 02 02 «проектирование и производство программно-управляемых радиоэлектронных средств РЭС».....	277-278

МАШИНОСТРОЕНИЕ

Жихорев Е.О., Домкин И.Н. Программирование технологий резания и обработки конструктивных элементов в виде «карманов» деталей на фрезерных станках с ЧПУ при помощи САМ-систем	279-282
Домкин И.Н., Жихорев Е.О., Анисимов В.С. Анализ особенностей и возможностей различных систем автоматизированного проектирования технологических процессов в машиностроении	283-286
Корба К.В., Анисимов В.С. К вопросу о модернизации станочного оборудования для высокоскоростной обработки резанием	287-289
Скурят С.В., Рыктер К.О. Способы снижения напряжения сварного соединения	290-291
Вабищевич А.С. Вакансионный механизм залечивания трещин в твердом теле	292-295
Ласточкин А.Н. Проводящий композиционный материал	296-297
Гавриловец Д.А. Персональная станция переработки офисной макулатуры	298-299
Гавриловец Д.А. Совершенствование эффективности блока формования плотности листа персональной станции переработки офисной макулатуры	300-301
Цёма И.М. Современные методы защиты и окрашивание кузовов автомобилей	302-305
Хомченко В.И., Жаворонок Е.А. Альтернативные методы диагностирования двигателей автомобилей	306-310
Хомченко В.И., Жаворонок Е.А. Восстановление шаровых опор при помощи полиамида	311-313
Жерносек М.Ю. Восстановление деталей пластическим деформированием материала	314-315
Шевчёнок Е.А., Якимович Е.Ю. Функциональный тюнинг автомобилей с целью повышения топливной экономичности и улучшения экологических параметров	316-318
Шевченко Д.С., Войтехович В.В. Сравнительный анализ качества бензина на АЗС города Новополоцка	319-320
Шевченко Д.С., Войтехович В.В. Анализ сети АЗС В Беларуси	321-322
Иванов А.Л. Оценка воздействия выбросов автомобильного транспорта на качество атмосферного воздуха в городе Новополоцке	323-324
Иванов А.Л. Проблема загрязнения атмосферного воздуха выхлопными газами автотранспорта в Республике Беларусь	325-326

ХИМИЧЕСКИЕ ТЕХНОЛОГИИ

Рябухин Р.М., Корбут Е.Я. Исследование свойств серо- и фосфорсодержащих присадок к моторным маслам с улучшенными химмотологическими свойствами	327-328
Рябухин Р.М., Корбут Е.Я. Экономическая эффективность внедрения новых технологий производства серо- и фосфорсодержащих присадок к моторным маслам с улучшенными химмотологическими свойствами	329-330
Сороговец М.В., Лапина Ю.В. Получение белого масла адсорбционной очисткой продуктов фракционирования остатка гидрокрекинга	331-334
Сороговец М.В., Лапина Ю.В. Оптимизация режима адсорбционной очистки масла гидрокрекинга	335-337
Поляшкевич П.М., Корбут Е.Я. Сравнительный технико-экономический анализ процессов получения синтез-газа	338-339
Коляго А.С. Технология производства, экономическое сравнение использования биогаза с имеющимися видами топлива для автомобильных двигателей	340-342
Коляго А.С. Экономическое сравнение установок по производству биогаза с альтернативными источниками для выработки электроэнергии	343-344
Поляшкевич П.М., Корбут Е.Я. Основные технологии получения синтез-газа и его место в современном мире	345-347
Туровец Е.А. Определение фракционного состава биодизельных топлив и их смесей	348-350
Туровец Е.А. Пирометрические методы контроля кипения в современных автоматических анализаторах фракционного состава светлых нефтепродуктов	351-353
Журидов А.М. Обзор и сравнение реакторов-электрокоагуляторов	354-356